UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS FÍSICAS

TESIS DOCTORAL

Hybrid Quantum-Classical Algorithms

Algoritmos híbridos cuantico-clásicos

MEMORIA PARA OPTAR AL GRADO DE DOCTOR EN FÍSICA

PRESENTADA POR

Roberto Campos Ortiz

Director

Miguel Ángel Martín-Delgado Alcántara

Madrid

Hybrid Quantum-Classical Algorithms

Algoritmos híbridos cuantico-clásicos

Roberto Campos Ortiz

Director: Miguel Ángel Martín-Delgado Alcántara

Facultad de Ciencias Físicas

Universidad Complutense de Madrid

Memoria para optar al título de

Doctor en Física

Febrero 2024

Acknowledgements

Una tesis doctoral queda lejos de la mayoría de mi familia y amigos. Así que estos agradecimientos no se los dedico por haber contribuido al proceso concreto, académico y sesudo, sino que me gustaría agradecerles haberme ayudado, en estos 31 años, a llegar a ser la persona que ha podido terminar esta tesis.

Me gustaría agradecerles a mis padres, Cory y Julio, haberme criado con amor, cariño, dedicación, inteligencia, sabiduría y una larga lista más de adjetivos positivos. Ellos me enseñaron a entender y amar la vida y a saber que no soy inferior a nadie. También fomentaron en mí la curiosidad y el placer del conocimiento, dos requisitos imprescindibles para hacer un doctorado. En resumen, lograron enseñarme la felicidad de ser uno mismo.

Mi agradecimiento más profundo es para Ángeles, quiero agradecerle todo, sin ninguna palabra más, porque cualquier palabra se quedaría pequeña para expresar nuestra relación. Estoy orgulloso de la persona que soy y hay mucho de ella en mí. Ignoro el futuro, pero sé que con ella, todo es mejor.

A Miguel Ángel Martín-Delgado, mi director de tesis, quiero agradecerle, primero, haber sabido explicarme con paciencia y entusiasmo los fundamentos de esta ciencia. Y, segundo, haberme enseñado cosas tan valiosas como liderazgo, perspectiva y, espero, cierta clarividencia para entender un mundo tan cambiante como el de la computación cuántica. Este agradecimiento es extensible a Carmen Page, nuestro apoyo en la sombra científica, siempre pendiente de nosotros para lo que haga falta.

Otra persona que no puede faltar en estos agradecimientos es Aitor Ibarra, mi otro director de tesis, más moral que académico. Mi primer contacto con Quasar y un apoyo constante durante la tesis. Me quedo con su incansable energía para sacar proyectos adelante contra todo y todos, aunque fuesen de alto riesgo. Ha sido un referente de cómo la tenacidad puede convertir una quimera en realidad.

También quiero agradecer a dos empresas su apoyo. Primero a Quasar, la otra mitad de este doctorado industrial, por entender que la investigación también puede ser rentable. Segundo a Zapata Computing, especialmente al que fue todo el grupo de QML y a Alejandro Perdomo como cabeza visible. El tiempo que estuve con ellos me sentí uno más de la familia, llevándome a partes iguales conocimiento científico y aprendizaje vital.

Por último, pero no por ello menos importante, quiero agradecer a toda la gente del departamento. Con ellos he compartido el día a día de este doctorado, esas personas con las que la rutina del trabajo siempre es agradable. No puedo poner la lista con todos los nombres que debería, pero sí mencionar algunos en especial. A Pablo Moreno, mi compañero durante una parte importante del doctorado, por enseñarme tanto con tanta paciencia. A Gabriel Escrig, por ser compañero y amigo, por el trabajo y las horas juntos, por la física y por la filosofía compartida. A Sara Giordano, por ser amiga y guía, por las muchas horas de charla y risas, tanta investigación como amistad. Y a Pablo Rabán, por ser un compañero de despacho que se convirtió en amigo, siempre atento, con una sonrisa, una canasta o un chiste, por ser tan grande.

Quiero incluir, en estos agradedicimientos, una última reflexión. Habrá gente que no aparezca explícitamente, pero, si en estos 31 años, has pasado tiempo conmigo y me has visto sonreir, de una forma u otra, el tiempo juntos me ha ayudado a ser como soy, por eso, gracias a ti también. Gracias a todos.

Abstract

Hybrid algorithms combine classical and quantum computing to create a whole that improves the performance of classical algorithms. There are many ways to understand this combination, in this thesis two are studied. First, it is an approach that combines classical and quantum modules communicating with each other to create a search & sample optimization algorithm. The second one is a classical algorithm that studies the cost and performance of quantum algorithms applied to chemistry.

Hybrid algorithms are interesting due to the limitations of both classical and quantum computing. Some hard problems solved by classical computing are limited in performance and scalability because due to the exponentially growing number of states to be evaluated. On the other hand, currently, quantum computing is limited because quantum hardware is not yet developed enough to execute algorithms that process large amounts of data. By combining both technologies, it is possible to obtain an algorithm capable of processing large volumes of data with greater speed.

The first proposed algorithm in this work, quantum Metropolis Solver, QMS, is an adaptation of a quantum walk to a quantum Metropolis-Hastings algorithm, applied to problems of industrial interest such as artificial intelligence, space exploration, energy sector or quantum chemistry. In these use cases, QMS has proven to have an advantage over its classical counterpart, being tested on both simulator and quantum hardware.

The second proposed hybrid algorithm, TFermion, follows a different paradigm. It is a classical algorithm that analyzes the cost of T-type gate of quantum algorithms applied to quantum chemistry. Knowing this cost is crucial to compare the ability of algorithms to be executed on real quantum hardware. The main algorithm of TFermion is to generalize this calculation to other methods and apply it to any molecule. Then, it has been applied to a real problem, the design of more efficient electric batteries.

Publications, Industrial Collaborations and Conferences

P1* R. Campos, P.A.M. Casares, M.A. Martin-Delgado, *Quantum Metropolis Solver: A Quantum Walks Approach to Optimization Problems*, Quantum Machine Intelligence **5** 00119 (2023) [1]

P2* P.A.M. Casares, R. Campos, M.A. Martin-Delgado, *QFold: quantum walks and deep learning to solve protein folding*, Quantum Sci. Technol. **7** 025013

P3* P.A.M. Casares, R. Campos, M.A. Martin-Delgado, *TFermion: A non-Clifford gate cost assessment library of quantum phase estimation algorithms for quantum chemistry*, Quantum **6** 768

P4 G. Escrig, R. Campos, P.A.M. Casares, M.A. Martin-Delgado, *Parameter estimation of gravitational waves with a quantum metropolis algorithm*, Class. Quantum Grav. **4** 045001 (2023)

P5 G. Escrig, R. Campos, P.A.M. Casares, H. Qi, M.A. Martin-Delgado, *Quantum Bayesian Inference with Renormalization for Gravitational Waves*, `https://arxiv.org/abs/2403.00846`.

P6 A. Delgado, P.A.M. Casares, R. Dos Reis, M. Shokrian Zini, R. Campos, N. Cruz-Hernandez, A. Voigt, A. Lowe, S. Jahangiri, M. A. Martin-Delgado, J. E. Mueller, J. M. Arrazola, *Simulating key properties of lithium-ion batteries with a fault-tolerant quantum computer*, Phys. Rev. A **3** 032428

***** Publications that the original paper appear in the thesis.

Awards

AW1 **3MT**: Accesit award in the contest 3 Minutes Thesis contest organized by Universidad Complutense de Madrid. 3MT is an outreach competition in which Ph.D. students had to explain their research in three minutes to a non-expert audience with only one slide, from a static position.

[1] IMPACT FACTOR (IF) 2024: Quantum Machine Intelligence. **IF: 5.64**, Class. Quantum Grav. **IF: 3.52**, Quantum Sci. Technol. **IF: 6.7**, Quantum. **IF: 6.4**, Phys. Rev. A. **IF: 2.408**

IC1 Company QuasarS.R.: Quasar is the industrial collaborator associated with this thesis through the framework of an Industrial PhD program supervised by Aitor Ibarra and Ignacio de la Calle. Quasar is a company operating within the space sector, with a specialized focus on providing artificial intelligence solutions for the European Space Agency.

IC2 Company Zapata Computing: An internship took place at this company from June to November 2022. This engagement was with the quantum artificial intelligence team, led by Alejandro Perdomo-Ortiz, and focused on an in-depth exploration of quantum generative models. Zapata Computing is a specialized company in the development of industrial-level quantum software.

IC3 Company Xanadu: This collaboration arose from the publication 8.4 in conjunction with the quantum algorithms and chemistry group at Xanadu, led by Juan Miguel Arrazola. The outcome of this collaborative effort, which also involved the automobile company Volkswagen, led to the publication of [1]. Xanadu is a comprehensive quantum computing company engaged in the development of hardware (HW) and software (SW) solutions.

IC4 LIGO Project: The partnership with the LIGO collaboration and Queen Mary University of London began due to their interest in the publication of [2]. The main goals were to improve the quality of the data and employ data representation methods similar to those used by LIGO researchers. LIGO's focus involves building advanced interferometers for measuring and analyzing gravitational waves. This collaboration ends with publication of [3].

IC5 Company Repsol: Collaboration with this corporation was initiated after their initial interest in the 4.3 publication. The core focus of this collaboration is the formulation of a quantum algorithm for the optimization of green hydrogen production. Repsol is a prominent Spanish conglomerate specializing in the manufacturing and distribution of fuels.

IC6 Company Amazon-AWS: Collaboration with the quantum computing team at AWS, QFold library, publication 7.3 was integrated as a case study for drug design. AWS provides an industrial platform for the execution of quantum algorithms designed by various entities. Their choice to spotlight QFold stems from its perceived relevance to companies and hospitals engaged in cancer treatment research.

IC7 Nebrija University: This affiliation emerged as a result of the industrial component of this doctoral endeavor, Quasar. Nebrija University expressed interest in establishing a Spanish startup network encompassing research domains linked to quantum computing. This connection culminated in diverse collaborative activities, including didactic assistance. Nebrija University stands as one of the pioneering Spanish institutions to introduce a dedicated master's program exclusively focused on quantum computing.

IC8 Wikimedia Foundation: Wikimedia Foundation created a program to develop Wikipedia content. The objective was to identify ten Wikipedia entries on a specific topic that had not already been created and have five students develop those entries under the supervision of an expert. Ten entries on quantum computing were chosen and supervised within this thesis.

C1- ICE-6 (**poster**), May, Online, 2020

C2- Quantum Techniques in Machine Learning, November, Online, 2020

C3- Quantum Chemistry with Qiskit, November, Online, 2020

C4- IEEE Quantum Week (**seminar**), October, Online, 2021

C5- Quantum Matter (**talk**), November, Bilbao, Spain, 2021

C6- QHACK, February, Online, 2022

C7- APS March Meeting (**talk**), March, Chicago, USA, 2022

C8- EQAI Summer School (**poster**), September, Udine, Italy, 2022

C9- QHACK, February, Online, 2023

C10- APS March Meeting (**talk**), March, Las Vegas, USA, 2023

C11- Quantum Matter (**poster**), May, Madrid, Spain, 2023

C12- HDCRS Summer School, May, Reikiavik, Iceland, 2023

C13- Tech Talk European Space Agency, (**talk**), June, Online, 2023

C14- QTYR, July, Madrid, Spain, 2023

C15- QTS12 (**talk**), July, Praga, Czech Republic, 2023

C16- Granada Seminar (**talk**), September, Granada, Spain, 2023

C17- MaDQuantum, (**poster and stand**), September, Madrid, Spain, 2023

C18- Quantum Techniques in Machine Learning, (**poster**), November, Geneva, Switzerland, 2023

Detailed summary in English

Introduction

The current situation of quantum computing is not much different from that of a giant with feet of clay. Without being able to demonstrate a practical quantum advantage, there is strong investment at both the industrial and academic levels to develop quantum algorithms. This giant, consisting of purely theoretical algorithms, promises of exponential advantages and infinite applications, sits on shaky ground with only small quantum devices incapable of sustaining the giant and muddied by the high processing power of classical computing. This thesis tries to combine mud and giant to strengthen the base of the structure, while sacrificing some height of the colossus.

Beyond metaphor, the point of view of this work is to try to design algorithms following a hybrid paradigm in which classical and quantum computing cooperate to solve the problem. Another strength of the algorithms presented is that the quantum modules have an approach closer to that of information processing than to that of problem definition by means of formulas and evolution. Due to this combination, algorithms that can work on industrially relevant problems following the paradigm of hybrid classical-quantum computing have been created.

This paradigm follows the philosophy of taking advantage of the benefits offered by both technologies and combining them. It is accepted that this combined architecture may not be the final solution for quantum computing, but it is an intermediate step that in the short and medium term can greatly benefit and accelerate the development of quantum computing, even allowing to have practical applications earlier than expected.

Part 1: Quantum Sample and Search (QS&S)

The first part explains the concept of search & sample algorithms as algorithms that can traverse a state space to find a distribution of the data or the state with minimum or maximum value. This type of algorithm is useful when it is difficult to analyze the state space with another method, and the only solution is to run an evaluation function to know the value of many or all of the states.

These algorithms are classically used to solve optimization problems and are the basis for other more complex algorithms, such as artificial intelligence algorithms. However, the scal-

ability of these algorithms is poor because they require visiting a high number of states and this number grows exponentially in the problems to be solved. Therefore, classically, there is a limitation that must be solved in a quantum way.

Quantumly, there are different ways to create a search & sample algorithm. In fact, there are several proposals in the literature. However, the ones that stand out above the rest and that are going to be used in this work are the quantum walks algorithms. These quantum walks have the ability to perform a search over a state space polynomially faster than their classical counterparts. On these, a quantum Metropolis-Hastings algorithm can be built that maintains the advantage, but with greater capabilities.

Using this quantum Metropolis-Hastings algorithm, an algorithm called quantum Metropolis Solver, QMS, has been built. QMS, the algorithm around which the whole part I is structured, is a hybrid algorithm that combines a quantum Metropolis-Hastings circuit with classical preprocessing of the problem to achieve quantum advantage in problems of industrial interest and reduced size. QMS is run on quantum simulators running on classical hardware to test its performance, but it has also been tested on quantum hardware to confirm the results obtained in the simulator.

QMS has the ability to solve any optimization problem formulated using a problem description and a state evaluation function. Therefore, it has been applied to different use cases such as hypothesis search in artificial intelligence, gravitational wave parameter estimation for space exploration or protein folding in quantum chemistry.

Part 2: Quantum Chemistry

The second part explores hybrid algorithms applied to quantum chemistry, specifically, to the prediction of the fundamental state of a quantum system and the calculation of its energy. In this case, the hybrid paradigm is applied from another perspective. Instead of having a quantum and a classical module connected to each other, this alternative has a classical algorithm that analyzes different quantum algorithms to estimate their cost in number of gates. In this way, it is possible to analyze what resources are necessary in a quantum device to be able to execute these algorithms, what difference there is between the available resources and the real needs, which approaches are more efficient, etc.

To do this, first, it is necessary to explain classical quantum chemistry algorithms and their limitations. These limitations can be overcome with quantum algorithms. However, these quantum algorithms are expensive, especially for current quantum hardware. Therefore, it is necessary to analyze the gate cost of each of the processes that perform the algorithms, the

Hamiltonian simulation, the encoding, the mapping, etc.

This analysis has been generalized for a set of quantum algorithms and any possible molecule in a software tool called TFermion. This tool analyzes in detail the T-gate cost of algorithms that are state of the art in quantum chemistry.

To show an use case of TFermion in the industrial environment, the problem of designing lithium-ion batteries has been chosen. The problem is focused on the cathode design. For this purpose, quantum algorithms can be used to estimate the fundamental state more accurately, and it is interesting to study the cost of quantum gates of these algorithms.

Resumen detallado en Español

Introducción

La situación actual de Computación Cuántica no difiere mucho de la de un gigante con pies de barro. Sin que se haya podido demostrar una ventaja cuántica en un problema real, ya se ha desarrollado una fuerte inversión tanto a nivel industrial como académico para diseñar algoritmos cuánticos. Este gigante, formado por, algoritmos puramente teóricos, promesas de ventajas exponenciales e infinitud de aplicaciones, se asienta sobre un terreno inestable en el que sólo existen dispositivos cuánticos de pequeño tamaño incapaces de sostener al gigante y embarrado por la alta capacidad de procesamiento de la computación clásica. Esta tesis trata de combinar barro y gigante para fortalecer la base de la estructura, amén de sacrificar cierta altura del coloso.

Más allá de la metafora, el punto de vista de este trabajo es tratar de diseñar algoritmos siguiendo un paradigma híbrido en el que computación clásica y cuántica cooperen para resolver problemas industriales. Otro punto fuerte de los algoritmos presentados es que los módulos cuánticos tienen un enfoque más cercano al del procesamiento de la información que al de la definición del problema mediante fórmulas y evoluciones. Gracias a esta combinación, se llega a algoritmos que pueden trabajar sobre problemas relevantes a nivel industrial siguiendo el paradigma de la computación híbrida clásico-cuántica.

Este paradigma sigue la filosofía de aprovechar las ventajas que ofrecen ambas tecnologías y combinarlas. Se acepta que esta arquitectura combinada puede no ser la solución final para la computación cuántica, pero sí un paso intermedio que a corto y medio plazo puede ser beneficioso y acelerar el desarrollo de la computación cuántica, permitiendo, incluso, tener aplicaciones prácticas antes de lo esperado.

Parte 1: Búsqueda y Muestreo Cuántico

En la primera parte de la tesis, se explica el concepto de los algoritmos de búsqueda y muestreo como algoritmos que son capaces de recorrer un espacio de estados para encontrar una distribución de los datos o el estado con mínimo o máximo valor. Este tipo de algoritmos son útiles cuando es dificil analizar con otro método el espacio de estados y la única solución es ejecutar una función de evaluación para conocer el valor de muchos o todos los estados.

Estos algoritmos son muy utilizados clásicamente para resolver problemas de optimización y son la base de otros algoritmos más complejos como los de inteligencia artificial. Sin embargo, la escalabilidad de estos algoritmos es mala porque requieren visitar un alto número de estados y, este número, crece de más rápido que la capacidad de cómputo de los ordenadores clásicos. Por ello, clásicamente existe una limitación que se intenta resolver de manera cuántica.

Cuánticamente, existen diferentes formas de crear un algoritmo de búsqueda y muestreo. De hecho, en la literatura, hay diversas propuestas. Sin embargo, la que sobresale por encima del resto y que va a ser utilizada en este trabajo son los algoritmos de caminos cuánticos. Estos caminos cuánticos tienen la capacidad de hacer una búsqueda sobre un espacio de estados polinómicamente más rápida que sus homólogos clásicos. Sobre estos, se puede construir un algoritmo de Metropolis-Hastings cuántico que mantiene la ventaja pero con mayores capacidades.

Utilizando como base este algoritmo de Metropolis-Hastings cuántico, se ha construido un algoritmo llamado Quantum Metropolis Solver, QMS. En torno a este algoritmo, se estructuran un conjunto de capítulos de esta primera parte. Se trata de un algoritmo híbrido que combina un circuito cuántico de Metropolis-Hastings con preprocesamiento clásico del problema para lograr ventaja cuántica en problemas de interés industrial y tamaño reducido. QMS se ejecuta en simuladores cuánticos que corren en hardware clásico para probar su rendimiento pero, también ha sido probado en hardware cuántico para confirmar los resultados obtenidos en el simulador.

QMS tiene la capacidad de resolver cualquier problema de optimización formulado mediante una descripción del problema y una función de evaluación de los estados. Por ello, ha sido aplicado a diferentes casos de uso como búsqueda de hipótesis en inteligencia artificial, estimación de parámetros de las ondas gravitatorias para la exploración espacial o el problema del plegamiento de proteínas.

Parte 2: Química cuántica

En esta segunda parte se exploran los algoritmos híbridos aplicados a la química cuántica, concretamente a la predicción del estado fundamental de un sistema cuántico y el cálculo de su energía. En este caso, el paradigma híbrido se aplica desde otra perspectiva. En vez de tener un módulo cuántico y otro clásico conectados entre ellos, se tiene un algoritmo clásico que analiza diferentes algoritmos cuánticos para estimar su coste en número de puertas. De esta manera, se puede analizar qué recursos son necesarios en un dispositivo cuántico para poder ejecutar estos algoritmos, qué diferencia hay entre los recursos disponibles y las necesidades reales, qué aproximaciones son más eficientes, etc.

Para ello, primero es necesario explicar los algoritmos clásicos de química cuántica y sus limitaciones. Estas limitaciones se pueden superar con algoritmos cuánticos. Sin embargo, estos algoritmos cuánticos son costosos, especialmente para hardware cuántico actual. Por ello, es necesario analizar el coste en puerta de cada uno de los procesos que realizan los algoritmos, la simulación del Hamiltonio, la codificación, el mapping, etc.

Este análisis se ha generalizado para un conjunto de algoritmos cuánticos y cualquier posible molécula en una herramienta software llamada TFermion. Esta herramienta analiza en detalle el coste en puertas T de los algoritmos que marcan el estado del arte en química cuántica.

Para mostrar un caso de uso de TFermion en el entorno industrial, se ha escogido el problema del diseño de baterías de litio-ion con el fin de mejorar su autonomía. El problema está centrado en el diseño del cátodo. Para ello se pueden usar algoritmos cuánticos que estiman con mayor precisión el estado fundamental y es interesante estudiar el coste de en puertas cuánticas de esos algoritmos.

Contents

Introduction

> The most beautiful thing we can experience is the mysterious. It is the source of all true art and science. He to whom the emotion is a stranger, who can no longer pause to wonder and stand wrapped in awe, is as good as dead —his eyes are closed.
>
> Albert Einstein, *Living Philosophies*

These words written by Albert Einstein reflect the reality of researching in quantum mechanics and by extension in quantum computing. This thesis became an exercise of dealing with the mysterious foundations of quantum computing from a different point of view from the current one, from the perspective of the theory of computation and algorithms. This new approach has been applied to the development of hybrid classic-quantum algorithms that solve current industry problems.

Quantum computing entails the application of quantum mechanics principles to the realm of information processing. Quantum mechanics, which ignited a revolution in 20th-century physics, challenged fundamental principles that had stood for centuries. Renowned physicists such as Max Planck, who introduced the concept of quanta through his exploration of black body radiation, Albert Einstein with his work on the photoelectric effect and extensive contributions to quantum physics, as well as notable figures like Louis De Broglie, Niels Bohr, Edwin Schrödinger, Max Born, Heisenberg, Dirac, among others, significantly advanced the field of quantum mechanics. This counter-intuitive and groundbreaking domain found practical utility in the creation of lasers, microprocessors, superconductors, and chemical models.

However, simulating the intricate quantum behaviors of a system using a classical computer proves to be extremely complex. Such simulations necessitate encoding intricate states and managing myriad interactions among them. This challenge led Richard Feynman to propose an unconventional solution: employing the principles of quantum physics themselves, leveraging quantum states to simulate quantum systems. Feynman, driven by the desire to expedite simulations, inadvertently laid the groundwork for what now is recognized as quantum —an anticipated revolution poised to shape the 21st century. Feynman's proposal is only the tip of the iceberg of quantum computing, and he was not the first. In 1985, David Deutsch introduced the concept of quantum Turing Machine [4]. It was equivalent to the Turing Machine designed by Alan Turing [5] but it proved that it was possible to execute the same set of operations with

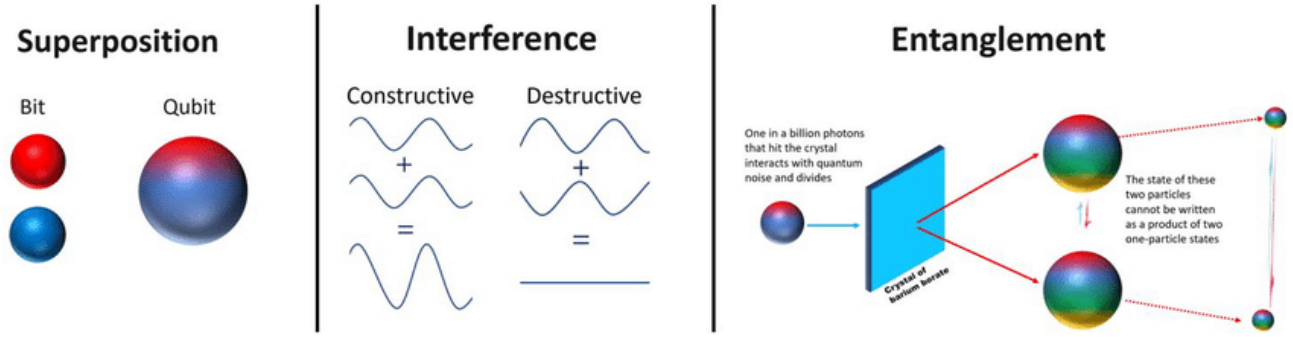

Figure 1: Scheme extracted from [7]. It represents the phenomenom of superposition in a qubit represented with two possible states, red and blue and a qubit that is a combination of both. The interference effect, in the middle, can be constructive if two waves interfer to create a pattern or destructive if both cancel each other. Finally, the entanglement can be undestood as two particles that can not be written as a product.

quantum hardware. Following the idea of simulating experiments on quantum computers, other scientists have proposed to process information faster and with lower energy costs to develop areas such as quantum chemistry, communications, or artificial intelligence.

If classical computing is conceived as the manipulation of 0 and 1 signals (indicating the presence or absence of electric current) and the implementation of logical gates that integrate these signals (AND, OR, NAND, XOR gates, etc.), Quantum computing shares similarities in terms of logic while diverging significantly in its properties. Quantum computing leverages quantum particles existing in superposition and entanglement states, enabling it to not only represent states but also to perform operations in an optimized manner compared to classical computing. The main similarity between both is that in classical and quantum computing the unit of information is the bit (or the qubit) with states 0 and 1. So, both use binary representation. Although it is important to note that this is just a simplification, it is sufficiently explanatory.

Superposition is the property that allows a quantum state to be in two states simultaneously. This implies that a qubit can exist in a state of 0, 1, or in a linear combination of both possibilities, resulting in a remarkably condensed information representation. Entanglement, on the other hand, is the phenomenon through which two quantum particles behave as though they are a single entity. In an entangled quantum system, any alteration that impacts the system equally affects both particles. The significance of two or more entangled qubits lies in their collective impact: operations (quantum gate) over the entangled system can affect all its qubits [6]. In Figure 1, these three concepts are visually represented.

Quantum computing offers numerous advantages over classical computing: superdense encoding of information, transmission through secure channels, reduced energy costs, operations

in superposition, etc. This thesis will focus on the ability to operate on a superposition of states, which allows for the reduction of computational complexity in algorithms [8].

All these advantages have strong theoretical underpinnings. There are works in computational complexity analysis that have created new classes like BQP [9]. Other works have formulated the quantum Turing machine [4] and derived the set of universal gates for quantum computation [10]. There are also studies that demonstrate polynomial or superpolynomial advantages over classical algorithms [11]. However, it is crucial to highlight that all these demonstrations are purely theoretical and have not yet been proven in quantum algorithms executed on real hardware. This means that, at this moment, quantum algorithms do not have an advantage over a classical one in solving interesting problems.

At this juncture, it is worth asking why quantum computing has not yet solved real-world problems. There is a simple answer, there are many limiting factors need to be analyzed.

- **Hardware**
  - *Quantum HW size (number qubits)*: The number of qubits is often considered the primary metric for evaluating the scale of a quantum computer. However, it is important to emphasize that this metric, while significant, is neither the most crucial nor the most constraining factor. Although some may draw parallels between the number of qubits and the storage capacity, RAM size, or CPU capabilities of a classical computer, such comparisons are not accurate.

    The number of qubits is directly tied to a quantum computer's capacity for representing different states. In simpler terms, a quantum computer with a higher number of qubits can represent a larger number of states, allowing it to address more extensive problem sizes. However, it is essential to recognize that the ability to solve larger problems does not necessarily equate to faster problem-solving.

    Instead, the true potential of quantum computing lies in its capability to tackle problems that classical computers may find unbeatable. It is also true that increasing the number of hardware noise qubits is moving towards the possibility of having logical noise free qubits which will speed up the quantum algorithms, but it is still a future scenario.
  - *Quantum Errors*: The execution of algorithms in a hardware device is never error-free, either classically or quantumly. Classically, these errors have been extensively analyzed and studied with different strategies to correct them. Quantumly, there are inherent errors in devices that make classical error correction techniques inapplicable. These errors are phase errors. This new type of error added to the quantum limitations of not being able to copy the states without altering them means that, at present, quantum

errors limit the execution of algorithms. Advances in quantum error correction will mark the ability to run longer and more complex algorithms.

- *Coherence times*: The duration during which a quantum system retains its quantum attributes, such as superposition and entanglement, is referred to as its coherence time. When a quantum computer has a limited coherence time, it implies that there is only a brief window available for executing quantum gates before the system's qubits lose their quantum properties. Consequently, enhancing the coherence times of quantum chips opens up the potential for executing more intricate and longer algorithms. Coherence times limit the circuit depth, which is the real limiting factor in the present quantum hardware [12].

- **Software**

  - *Algorithms*: The full harnessing of quantum computing's advantages remains a subject of ongoing exploration, and consequently, quantum algorithms have yet to achieve their maximal speed and efficiency. Quantum computing processes information in a manner so distinct than classical computing, that current efforts primarily involve attempting to replicate classical algorithms to ascertain any potential advantages. However, in the long-term trajectory, this strategy must evolve to encompass the development of entirely quantum algorithms that are not reliant on classical inspiration.
  - *Representations*: A significant constraint of quantum algorithms lies in how information is represented. Many of these algorithms attempt to encode information into a formula, rendering quantum processing similar to quantum simulation rather than information processing. It is imperative to recognize that in the term "quantum computing", the emphasis is more on "computing" than "quantum". In other words, the main focus should be on information processing.
  - *Operators*: A quantum operator is composed of a set of gates, and it is an extra layer of abstraction in the process of implementing a quantum algorithm. An operator, in the context of quantum computing, serves as a programmer's tool for executing specific operations in a manner that leverages the capabilities of a quantum computer. However, a significant challenge arises from the relatively limited number of available operators. In fact, many algorithms that theoretically exhibit a quantum advantage are derived from or heavily influenced by two operators: the Grover diffusion operator [13] and the amplitude estimation operator, famously employed by Shor in the realm of prime number factorization [14].

- **Inherent**

  - *No cloning theorem (information storage)*: While quantum computing presents transformative quantum properties, it also introduces challenging features that complicate

its manipulation. Perhaps the most crucial of these challenges is the fact that observing a quantum system inherently disrupts it. In practical terms, this means that quantum states cannot be duplicated without undergoing destruction, because the very act of examining the state unavoidably alters it. This principle is encapsulated by the non-cloning theorem and affects, currently prohibits, some crucial operations in computer science, copying and storing information. However, this problem can be avoided since quantum states can be stored in a quantum memory and retrieved by quantum teleportation, which does not violate the non-cloning theorem. On the other hand, this limitation becomes an advantage in the context of quantum cryptography, where not being able to copy a state guarantees that it is secret.

All these factors listed above are just an enumeration of the key points guiding the development of this new technology, the faster they evolve, the sooner quantum computing will become a reality over or alongside classical computing.

Currently, the development of quantum computing is a global and combined effort between academia and companies in which everybody is focus on simulation, hardware and software. This collaboration is almost complete and results in both, companies and universities are publishing their advances. It should be noted that companies are willing to collaborate for several reasons. First, there is no direct application of quantum computing to industry, so they do not yet have a clear and defined business. Second, there are still many challenges to solve and work on, so the relationships between the various players are still symbiotic. Finally, the volume of business offered by quantum computing is so large that a few companies are not able to cover it, and it is more profitable for them to collaborate than to compete. This also translates into a high shortage of workers in quantum computing.

Looking at the profile of scientists working in quantum computing today, most of them are physicists specialized in quantum physics who have made the step to quantum information for both quantum algorithms and quantum many-body systems. There are also many experimental physicists working on the development of quantum computers. Looking for other profiles working in quantum computing, most of them are engineers working on the adjacent circuitry in quantum chips.

However, in the field of algorithms and software development, the number of computer scientists is minimal. This is understandable because the first steps of quantum computing had to be developed by those who understood quantum theory with great precision, and that required extra effort from computer scientists. Today, that situation has changed; It is already feasible to devise and implement algorithms with a certain level of complexity, and the simulation-oriented approach traditionally employed by physicists in algorithm design is proving to be inefficient. For this reason, companies and universities are increasingly demanding physicists who are ca-

pable of programming and designing algorithms, or even computer scientists with a background in quantum physics.

This thesis, among its various contributions, endeavors to introduce a distinct perspective to the development of quantum algorithms. Rooted in the experiences of someone accustomed to working with data processing algorithms and classical artificial intelligence, it advocates an approach centered on information processing rather than the mere simulation of physical phenomena. Although this statement may appear somewhat ambiguous, subsequent sections will elucidate the development of algorithms that emphasize the representation of information through variables, with diverse values giving rise to a multitude of states. Subsequently, there will be an effort to combine these variables with specific problem rules to derive solutions. This approach aligns more closely with Grover's philosophy of state processing than with Shor's emphasis on phase estimation or adiabatic computation grounded in Hamiltonian evolution.

Following this approach of creating quantum algorithms with a stronger focus on solving industrial problems from a more computer-centric perspective, this doctoral research program has been developed. It has earned the distinction of an industrial doctorate. This recognition acknowledges the ability of this work to provide solutions originating in academia to the industrial domain, bridging the gap that often exists between these two realms. This distinction is particularly apt, considering the past decade has witnessed an explosion of startups following the spinoff model. In this model, expert scientists from various universities have developed prototypes that are both marketable and valuable to industry.

In the specific case of this work, the Spanish company Quasar identified quantum computing as a promising line of business for its interests. Quasar is an aerospace company that develops artificial intelligence and optimization algorithms applied to data analysis and spatial solutions. One of Quasar's main customers is the European Space Agency, for which it has developed products such as autonomous analysis of satellite images for the observation and monitoring of the Earth or an autonomous robotic telescope with the capacity to manage by itself the observations of a night in different hemispheres.

Quasar understood that the complexity and volume of space data increasingly required more computing power, especially when the space sector has just entered the so-called new space era, with private companies licensed to launch and operate satellites [15]. This led the company to look for new solutions to overcome the bottleneck of data processing and optimization algorithms. To this end, an industrial doctorate was proposed in collaboration with the GICC group of the Complutense University of Madrid and was awarded the Industrial Doctorate scholarship of the Community of Madrid (IND2019/TIC17146).

In order to carry out this industrial doctorate and to be able to create interesting algorithms for industry and collaboration with leading companies, during the development of this thesis,

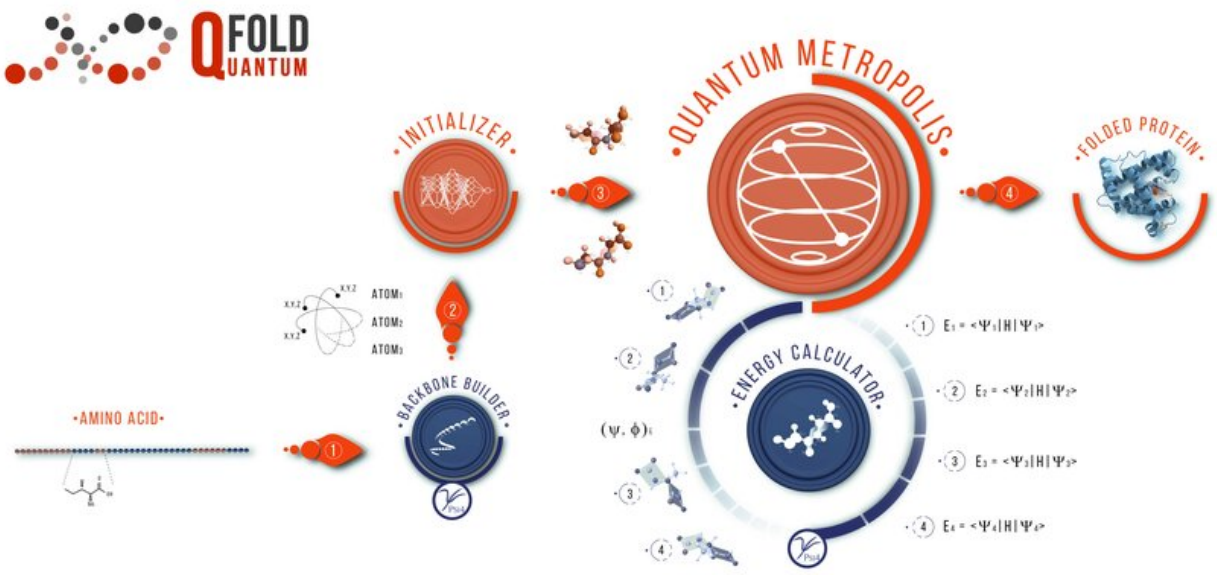

Figure 2: Scheme extracted from [16] representing a hybrid algorithm. Red modules are quantum and classical ones are in blue. Quantum and classical modules cooperates to get a solution. The system started with a chain of aminoacids with all angles initialized to 0. Then, an initial guess of the angles is done in the initializer and this initialization is refined in the quantum metropolis module. The energy calculator in blue is a classical module to calculate the energies of each possible configuration. Finally, the folded protein structure is obtained.

the main lines of research being pursued by other companies and universities have been analyzed in detail.

As discussed above, quantum computing offers state processing capabilities far superior to classical capabilities. However, it has been analyzed that quantum computing still has numerous problems to be solved. Some of these problems are particularly well solved by classical computing. For this reason, some years ago scientist started to discuss about hybrid classical-quantum algorithms. These algorithms try to combine the advantages of both worlds, the large data handling and storage capacity of classical computers and the superposition processing of quantum computers.

Hybrid algorithms consist of architectures that include modules from both technologies. Typically, a first module for classical data preparation and processing, then a quantum analysis module, and finally a classical module to analyze the output of the quantum module as it can be seen in Figure 2.

Following this architecture, hybrid algorithms offer numerous advantages in the short term.

- **Good points of both worlds**: Hybrid algorithms combine the fast analysis and preprocessing of data from classical computing with the ability to perform superposition processing and quantum operations from quantum computing.
- **Speed up against classical**: Hybrid algorithms, which incorporate a quantum module, perform intricate operations of classical algorithms with a notable speedup compared to purely classical algorithms.
- **Execution on real hardware**: The limitations of quantum hardware mentioned earlier may prevent the execution of a complete quantum algorithm. However, contemporary quantum chips are sufficiently capable of executing certain components or segments of these algorithms. This capability allows for the execution of hybrid algorithms on real quantum hardware.

These hybrid algorithms fit perfectly with the current approach to quantum computing. QC development has already been discussed to help overcome the limitations of classical computing. However, classical computing works at many levels, from personal computers, office systems, to computing centers, and high performance computing (HPC) [17].

The fundamental goal of QC development is to assist or replace those supercomputers used in HPC. This is due to several reasons, first because finding quantum advantages with current devices requires problems of high complexity, something that a personal computer will never solve, second because current classical computing is both effective and efficient at solving simple problems, third, quantum computers must be controlled by experts, something that makes their mass production and maintenance difficult, and finally, because the cost of a quantum computer must be borne by a large corporation.

The current short- and medium-term approach may be modified in the long term. It would not be surprising to find applications of quantum algorithms for personal computers over time. For example, the scientists who operated the first room-sized computers never thought of mobile devices, virtualization, or social networks, but over time these were developed. However, it is clear that, should this development take place, it will be in the long term.

Research in quantum computing endeavors to construct quantum algorithms capable of addressing problems that are computationally intractable for classical algorithms. Many of these algorithms designed to tackle such challenging problems are currently running on supercomputers [18]. Consequently, in the near term, the development of quantum computing is geared toward either supplementing or potentially supplanting supercomputers in specific computational tasks. Nevertheless, a fundamental characteristic of supercomputing is its ability to handle large volumes of data, often referred to as big data. Quantum computers, on the other hand, face limitations when dealing with such extensive datasets. Therefore, in the present sce-

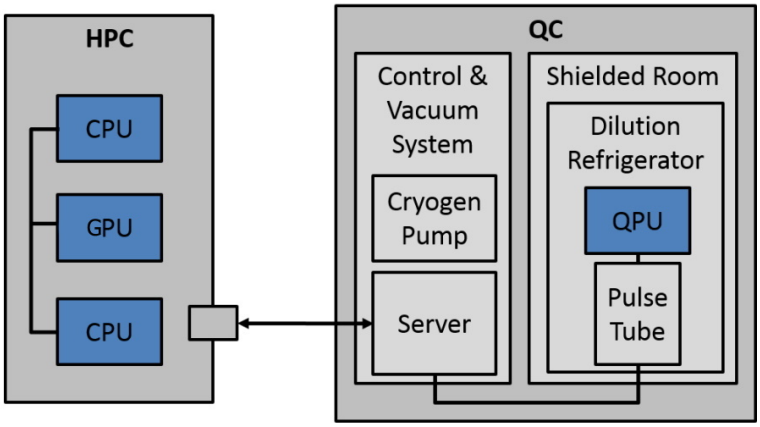

Figure 3: Scheme extracted from [21] representing a hybrid architecture integrating a QPU in a HPC system. It represents a possible architecture in a HPC pipeline, in which the quantum computing is integrated as one extra module to execute specific algorithms. Following the proposed architecture, QC module would be equivalent to GPU or CPU modules.

nario, the integration of both technologies becomes imperative, and hybrid algorithms emerge as a sensible approach to bridge the gap between these two realms.

Besides, different works [19] consider hybrid algorithms as a bridge that ends in pure quantum algorithms. Currently, it is not well known how to quantize some operations or how to get advantages from quantum computing in areas like quantum artificial intelligence (QAI) or quantum chemistry and the only solution is to apply quantum computing to some modules of the classical pipeline. This hybrid solutions will be an useful intermediate step to learn more about quantum information processing and to develop pure quantum algorithms.

On the other hand, there are authors who claim that quantum computing will be just an additional processing module within a large classical architecture [20]. This concept would be to have a quantum processing unit (QPU) inside a classical computer equivalent to current graphics processing units (GPU) as shown in figure 3.

Whichever view of hybrid algorithms prevails, it will be a battle that will be fought over the medium to long term. In the short term, hybrid algorithms will continue to develop because it is the fastest way to the concept of quantum advantage, understood as an algorithm that uses quantum computation and is faster than a classical one. Gradually, better interfaces between classical and quantum modules will be developed, optimizing the encoding of information and optimizing its execution.

Hybrid algorithms also hold significance due to their capacity to handle varying data types,

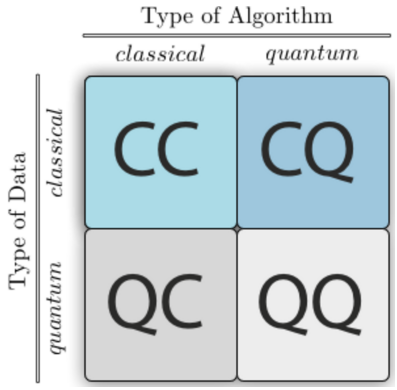

Figure 4: Scheme extracted from `https://en.wikipedia.org/wiki/Quantum_machine_learning#/media/File:Qml_approaches.tif`. In this figure is possible to observe four different paradigm for algorithms attending to the type of algorithm and type of data. If the algorithm is classical and also the data, the algorithm is similar to any existing algorithm in classical computing. The option of a quantum algorithm working with classical data is similar to the quantum algorithm proposed in this chapter applied to optimization problems. If the algorithm is classical but the data is quantum could be the classical simulation for quantum physics. Finally, if both, data and algorithm, are quantum, would similar to a quantum algorithm working on quantum chemistry. This paradigm will be explored in part II.

encompassing both classical and quantum data. As mentioned earlier, quantum computing was initially proposed primarily for quantum simulation, yet it swiftly emerged as a valuable tool for a broader spectrum of algorithms. Hence, based on the nature of the information involved, a classification can be drawn, distinguishing between classical or quantum data and classical or quantum algorithms, resulting in four possible categories, as shown in figure 4.

Hybrid algorithms are capable of working in three of these four categories, such as classical algorithms with quantum data [22], quantum algorithms with classical data [2] or quantum algorithms with quantum data [23].

The ability to run quantum algorithms on real hardware represents one of the significant advancements in the field of quantum computing in recent years. This milestone owes its thanks to the advent of noisy intermediate-scale quantum (NISQ) computers. These NISQ computers are compact devices comprising 10 to 1000 qubits, characterized by considerable noise levels, but capable of executing small-scale algorithms. They serve a dual purpose as a test-bed for algorithm development and as an intermediate step towards achieving fault-tolerant algorithms, where the anticipated quantum advantage is expected to shine truly [24].

NISQ hardware development has created an industrial opportunity for many companies. Big companies like IBM, Google and Microsoft started the research in this technology and have made great strides in quantum computing in both hardware and software, strongly supported

by research conducted at universities. This juncture was the breeding ground for the emergence around 2015 of small startups led by scientists coming from large universities and financially supported by rounds of funding and venture capital. These startups have continued the development of all QC research lines following a more academic than industrial model.

This diversity of actors and their collaboration is mainly due to the lack of competition and the early stage of the development of the technology. A paradigmatic example of this situation is that the three main proposals that have demonstrated quantum advantage over an artificial problem:

- **Google (2019)**: Using Sycamore chip, they solved a sampling random circuits problem faster than a classical computer [25].
- **Chinese University of Science and Technology (2021)**: Using the Jiuzhang chip, they solved a problem of gaussian amplitude sample faster than a classical computer [26].
- **Xanadu (2022)**: Using the Borealis chip, they solved a Gaussian boson sampling problem faster than a classical computer [27].

These three experiments were useful in demonstrating that quantum computing can solve certain problems faster than classical algorithms. However, they have no real applications. The interesting lines with real applications in which quantum algorithms are expected to be developed in the future are:

- **Optimization**: Problems such as how to deliver packages in a city while minimizing the number of trucks or the fuel used are of enormous relevance and can be interesting for QC because they encompass many possible combinations to check.
- **QAI**: Problems such as classification, speech recognition, image generation, or prediction are presently addressed by classical computers, although with certain inherent limitations. To tackle these problems at scale, a technology like quantum computing, which leverages operations in superposition, becomes essential.
- **Quantum chemistry**: Classical computers often expend significant resources when encoding a chemistry simulation. In contrast, quantum computers inherently represent chemistry quantum systems, enabling data processing with far greater efficiency and precision.
- **Quantum simulation**: Unlike quantum chemistry, quantum simulations will have substantial benefits from quantum computing. This is because executing operators that are inherently present in the quantum circuit enhances both the precision and the scale of systems that can be effectively simulated.

- **Communications**: The paradigm of quantum mechanics opens opportunities for improving classical communication methods by providing secure channels, superdense encoding, and speedy communication.

However, all of these lines of research must be accompanied by a strong development of quantum hardware. As seen before, the limitations of noise, coherence, and size of quantum computers affect the algorithms to the point of rendering them useless. Currently, NISQ computers are adequate as proofs of concept, but if quantum computing is to be used in real environments, a transition to a more robust quantum chip paradigm must be made. Fault-tolerant computers, capable of correcting their own errors and increasing coherence times, have emerged. Moreover, even these robust quantum computers seem to be assisted by classical computers, ie hybrid algorithms [28], which is a further motivation for this thesis. At least this is how research is developing at the moment.

Throughout this thesis, hybrid algorithms that address some of the main problems of interest to the industry have been explored. The interest was not only to develop algorithms to solve some of the problems in these areas, but also to better understand the concept of information processing in quantum computing and to create algorithms that were not only theoretical but had code implementation and the ability to be executed in simulators or quantum hardware. Therefore, each of the works explained has an associated code that can be executed. The topics that this thesis has delved into are optimization, quantum artificial intelligence, and quantum chemistry, which are three of the main ones listed above.

This thesis is organized as follows. The material has been divided into two main parts: (I) Part I on the work on search & sample algorithms very used in optimization and quantum artificial intelligence; (II) Part II on the work done analyzing algorithms of quantum chemistry and their relevance to the battery sector. Each part is divided into chapters.

Chapter 1 provides a concise introduction to the concept of search & sample techniques in quantum algorithms.

Chapter 2 provides a review of the classical algorithms already implemented for search & sample.

Chapter 3 provides a review of the quantum algorithms already proposed for search & sample.

Chapter 4 provides a review of an algorithm for applying search & sample to optimization problems. This chapter is based on the publication 4.3.

Chapter 5 provides an explanation of how search & sample can be applied to quantum artificial intelligence.

Chapter 6 provides an explanation of how search & sample can be applied to solve problems in the Space industry. This chapter is based on the publications [2, 3].

Chapter 7 provides an explanation of how search & sample can be applied to solve problems related with biology, in particular, the protein folding problem. This chapter is based on the publication 7.3.

Chapter 8 provides a brief introduction to Ground State Preparation (GSP) techniques. It also includes a detailed explanation about the TFermion software tool. This chapter is based on the publication 8.4.

Chapter 9 provides an explanation of how GSP techniques can help in battery design. This chapter is based on the publication [1].

Part I
Quantum Search and Sample (QSS)

1

Introduction to S&S

Searching and sampling techniques encompass a family of algorithms that can be described through various definitions. In order to present these concepts in simple terms:

- **Sampling** is a technique to approximate an unknown distribution defined by a function. This can be likened to a plumber who needs to understand the shape and path of a pipe hidden inside a wall. To achieve this, the plumber makes small openings in the wall and examines the pipe at various points, gradually gaining a sufficiently accurate understanding of its shape to perform the necessary work.
- **Searching** is a technique that involves finding a specific element within a distribution, typically the maximum or minimum, by making queries to the underlying function. Comparatively, this is similar to the plumber's task of locating a hole or issue in the pipe by tracing the flow of water within the pipe. Effective searching benefits from prior sampling, as it provides information about the distribution being searched.

These analogies illustrate the fundamental concepts of search & sample techniques in a more accessible way. In the realm of quantum computing, algorithms based on search & sample are developed to address optimization problems efficiently, often surpassing the capabilities of classical methods.

In this thesis, a search & sample technique is defined as a iterative method that allows to perform a search to find a reduced subspace of the complete search space and to sample that subspace to know its distribution. In the next step, using the information from the previous sampling, make a new search for another interesting subspace. By using this iterative methodology, it will be possible to know the global distribution of the problem to search in some cases for a minimum or maximum (pure search) or to sample the space.

In this work, the set of search & sample (S&S) algorithms are considered as a distinct category within the broader realm of techniques to solve optimization problems. While this perspective might be considered somewhat innovative, it is important to note that focusing specifically on search & sample algorithms, in reality, does not introduce anything fundamentally novel. Instead, this new category simplifies the algorithms, allowing to abstract certain features of optimization algorithms that are not essential for these problems.

The reason why these two types of algorithms can constitute a category of their own within optimization problem techniques is that they have many similarities and applications.

- **Numeric methods**: Search & sample techniques are numeric methods that calculate the solution with repeated queries to a function that returns a value. Both techniques refine the solution as more iterations of the algorithm are executed. Besides, it is important to notice that each iteration of a S&S algorithm is very simple to execute, so a problem can be solved efficiently executing many iterations in a row, as opposed to more precise and complex methods. [29]
- **Complex functions**: Search & sample techniques are specially useful with complex distributions that are difficult to analyze with other techniques. For example, functions that are costly to derive. The objective of S&S is to reduce the number of times that the function is executed, in particular if exhaustive search is the only known solution. [30]
- **Approximated methods**: Since they are approximate methods, in most of the cases, S&S return a solution that is not optimal, just a suboptimal approximation. For example, sampling a function will require to sample the same number of states that the function have to return an exact solution, which is not efficient. In the case of searching, it could return a maximum that is not the absolute minimum, but it is close enough.
- **Scalability**: S&S methods should work on large state space in order to get an advantage over other methods complex methods. Classically, this type of problem with large state space suffers from the curse of dimensionality [31], which results in a problem for classical algorithms due to its poor scalability. The number of times that a S&S algorithm needs to call the function tend to scale exponentially when the problem grows linearly in the number of states [32].
- **Closely related**: Search & sample algorithms are closely related because combining them could be very useful in some problems. For example, performing a search task is much simpler if a good sample was taken prior to the problem [33].

During the development of this thesis, it has been observed that search & sample algorithms are especially effective in optimization problems that meet certain characteristics. In this work,

optimization algorithms are understood as those in which it is necessary to find a solution that meets certain criteria and is of maximum reward or minimum cost, depending on whether one is working on a maximization or minimization problem, respectively.

The main characteristics that an optimization problem must meet for S&S algorithms to gain any quantum advantage is that it requires an almost exhaustive search in the state space to find the solution, and that the state space is large or grows almost exponentially. Although these may seem like very detailed or specific features, they are common in optimization problems [34].

If solving an optimization problem requires the use of algorithms and computers, it means that it involves so many variables, that the relationships between them are considerably large and that the evaluation function of the problem is complex to evaluate. Therefore, every time the problem changes or grows, the amount of resources to solve it will also increase. In addition, optimization algorithms, since they require finding the best solution from among many possible solutions, it is necessary to try almost all combinations between nodes until the optimal or at least a suboptimal one that is close to the solution is found. Therefore, in many optimization algorithms, the best technique for finding a solution is to search for possible combinations. Moreover, this search is almost exhaustive because it is very difficult to say which combinations of nodes should not be tried because they will not improve the quality of the solution.

The complexity of an algorithm that must do an exhaustive search on a problem grows exponentially. This is because each time a new variable is added, the number of new states grows by combining that new variable with all existing ones. This implies that the scalability of the algorithm is bad, because by slightly increasing the size of the problem, it becomes intractable [35]. This poor scalability is mainly due to the fact that classical algorithms evaluate states individually. Quantum, on the other hand, evaluates a superposition. Thus, the quantum algorithm will increase its complexity logarithmically in the representation of the states, but it will still always act on a superposition of the state space and its scalability will be better than that of the classical algorithm, always remembering that this applies to exhaustive search algorithms [36]. For example, in the quantum S&S algorithm is based on Grover, the advantage will up to polynomial.

It is at this point of poor scalability of classical algorithms, that quantum S&S algorithms appear, with an expected advantage in their scalability. The strength of these algorithms comes from the theoretical advantages obtained by Grover's operators and their different variations [37]. The main concept for understanding why quantum algorithms can accelerate the solution of optimization problems using QS&S algorithms is the application of operators in superposition.

To understand this concept, it is necessary to think that classically, the difficulty of the processing consists of evolving the states until finding the one with the minimum cost. Quantumly, the difficulty lies in increasing the probability of the minimum cost state. To evolve the

classical state, it is necessary to apply the operators individually and sequentially. However, the modification of the quantum probabilities can be done over the entire superposition, with a single operator simultaneously affecting all possible states. If the number of operations to increase the probability of the quantum state is less than the number of operations to evolve the classical state, then there will be a quantum advantage in the algorithm.

Finding optimization problems in which an almost exhaustive search must be performed and, therefore, are likely to have a quantum advantage is not complicated. Classically, computer scientists have identified some of these problems as very representative and have tested different algorithms on them. Then, when they wanted to solve other problems efficiently, they adapted them to be as similar as possible to the benchmark problem and thus applied the same algorithm. Even going a step further, they defined certain toy problems that were a simplification of real problems, but faithfully reproduced the complexity of the problem, making it very easy to go from the theoretical problem to the real problem. Some of the theoretical problems that were proposed according to the category of problems they solved were as follows:

- **Theoretical routing problem:** Problem of finding the best route between two cities. The test bed problem proposed is the Travel Salesman Problem (TSP) where a traveler starts in one city, visits a series of cities once, and ends up in the same city he started in [38, 39].
- **Theoretical configuration problem:** Problem of finding the best load balance for a small space. The proposed example problem was the knapsack problem. In this problem, the objective is to fill a knapsack that supports a given weight with a configuration of objects that maximizes a certain reward. [40, 41]
- **Theoretical artificial Intelligence:** In machine learning, there are many processes that need to be optimized, such as hypothesis search or hyperparameter optimization. Along these lines, problems such as N-Queen have been used to test algorithms that were then used in more complex techniques. The N-Queen problem consists of distributing n queens on a chessboard n by n in such a way that they do not attack each other [42].
- **Theoretical sampling:** Problem to characterize a function by taking only some samples of it. For example, to find the value of pi, it is possible to take a square and a circle such that the diameter of the circle is equal to the side of the square. By taking random samples and seeing which ones are inside the square and which ones are outside, it is possible to estimate a fairly accurate value of pi [43], as seen in Figure 5.

The important thing about these theoretical problems is their simple applicability to practical problems.

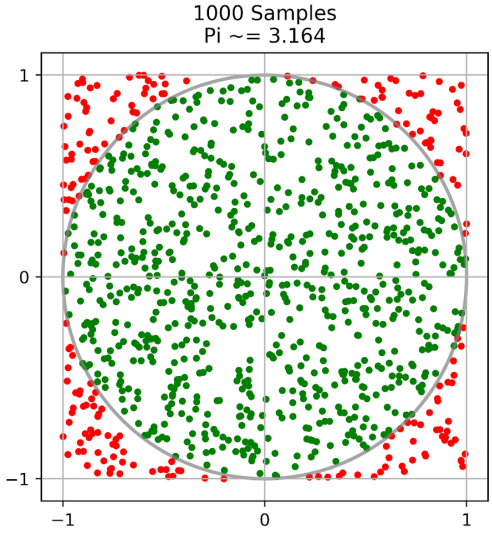

Figure 5: Pi is estimated using the number of green points that fall into the circle and the red points that fall out of the circle. This is a typical example to explain Monte Carlo simulation.

- **Routing problem:** The problem of finding a route between two or more points through a series of intermediate points is the problem solved by any package, courier or transport company to minimize the number of vehicles or the distance traveled. The larger the problem solved, the greater the number of points, vehicles, people or packages transported, etc., the greater the company's profit. There are numerous variants such as minimizing distance, time, fuel, etc.
- **Configuration problem:** The problem of delivering a number of items in a set of containers is a common one for packaging companies, international freight forwarders or almost any company in the industry known as last mile delivery. It is about cramming more products into a smaller space. Solving this problem more efficiently helps to reduce the number of containers or the size of containers, to give a few examples.
- **Artificial intelligence:** Optimization problems in artificial intelligence can range from selection of the hypothesis that best explains a data set, greater generalization and coverage with fewer clauses, to selection of hyperparameters that improve the performance of a neural network on a particular problem. For example, the unsupervised learning algorithm, k-means requires, in each iteration, to distribute centroids and calculate distances in such a way that the points are as well distributed as possible.
- **Sampling:** Sampling problems are widely used at the industrial level when it is difficult to know a certain distribution. For example, in the financial sector, to know the trend of

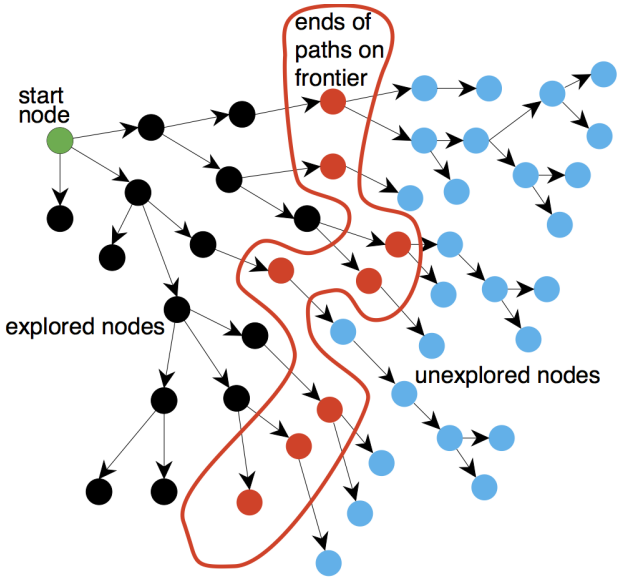

Figure 6: Scheme extracted from `https://artint.info/2e/html2e/ArtInt2e.Ch3.S4.html`. It shows the elements of a search from a start node. During the search, there are different categories of the nodes. In black, the already explored nodes which have been already evaluated and successor have been calculated. Then, in red the frontier nodes that are under evaluation in this step. Finally, the blue nodes have not been explored yet, the solution is one of these nodes.

an asset, or to improve risk analysis. In these cases, the ability to solve larger problems in shorter times allows obtaining more accurate short-term results.

So far the interest in using quantum S&S algorithms has been generally motivated, the next step is to study both techniques in more detail and explain the relevance of this type of algorithms as they fit into optimization problems.

1.1 Search

Search algorithms are techniques that visit nodes of a problem with a large search space until the state of minimum cost or maximum reward is found, as shown in figure 6. The main objective of these algorithms is to evaluate the minimum number of possible states until the solution is found. As with sampling algorithms, the search will always take a set of problem states and try to find a solution that is extrapolable to the entire problem.

Applying a search algorithm to a problem is interesting if the problem is large enough that evaluating all possible states or combinations is too costly. Thus, the fundamental objective

of a search algorithm is either to evaluate a subset of states that contain the solution to the problem (minimum cost or maximum reward state) or to evaluate the fundamental nodes that give it information on where the solution node is.

To simplify the search task, the problem can be represented in the most efficient way possible for the algorithm. In this way, there are representations of problems such as graphs, trees, constraints to satisfy, Stanford Research Institute Problem Solver language, STRIPS [44], etc. If search is understood as a problem in which an agent must find a solution state, an efficient representation may be to represent the problem as the union of states, actions and rewards. Thus, there would be a set of states S, a set of actions A, and a set of rewards R.

In this context, whether the search algorithm is understood as an agent or not, it must visit the states sequentially, one after the other, to evaluate each one and generate the successors. This process will then be repeated iteratively until the solution or a stopping criterion is found. This sequential analysis of the states is one of the most limiting factors for the feasibility of a search algorithm. Each time the problem grows, a new variable is added to the state space, the state space grows exponentially and, therefore, the number of states that need to be evaluated also increases.

Among the various techniques for speeding up search algorithms, search parallelization stands out. This optimization aims to ensure that the states are processed in parallel by n threads. While it is true that this is an advantage and is used, the computational cost remains high and scalability limited. [45, 46]. Other techniques to avoid sequential search are bidirectional search and random search, but they are still algorithms with low accuracy and some computational cost.

There are even problems in which the search to find the solution is practically exhaustive, i.e. it is necessary to visit all the possible states of the problem until the solution is found. To give a simple example, to find the correct combination of a lock, it is necessary to try all the possible solutions until the correct one is found, and by evaluating a state it is impossible to know how close or far it is from the solution. In cases where an almost exhaustive search is required, the simultaneous evaluation of nodes has the greatest impact.

Search algorithms can be divided into two main categories according to their choice of successors, informed or uninformed search. According to this classification, if the algorithm chooses the next node to be evaluated without having an estimate up to the solution node, it is called an uninformed search. If, on the other hand, among the possible nodes to be expanded, an estimate is made of the distance of each one to the goal and the one that is closest to the goal is chosen, it is called informed search.

Within an uninformed search, systematic search algorithms such as breadth-first, depth-first,

or bidirectional algorithms, or stochastic algorithms such as random walks can be distinguished. Uninformed search is fast because it does not need to evaluate the distance from the state to the goal, but it either has low precision, like a random search, or high cost, like a breadth-first search.

Informed search is much more accurate, but requires some domain knowledge to develop some kind of heuristics to guide the algorithm to the solution. In this category are heuristic search algorithms such as A* or alpha-beta pruning. In this type of algorithm, a balance must be struck between a sufficiently informative evaluation function and one that is fast to compute because it is to be evaluated at each node.

Search algorithms have numerous applications for problems where it is necessary to find a solution that meets certain constraints. These constraints can be optimality constraints, maxima or minima, restrictions, etc. For this reason, search algorithms are closely related to optimization problems. These problems involve numerous constraints, and it is necessary to try different configurations of parameters until one is found that satisfies all of them.

Some examples where search problems are useful are routing problems to find the best path among possible paths, configuration problems to find the combination that maximizes reward, or constraint satisfaction problems to find a particular configuration. In this context, artificial intelligence problems can be easily adapted to problems where the best technique to solve is search [30].

1.2 Sampling

Statistical sampling of a function is a technique that, from a set of data called population, selects a smaller subset called a sample. This sample is expected to be sufficiently representative to be able to extrapolate any analysis made on it to the population. In this way, working with the reduced sample will allow one to know a general and approximate trend of the population, but at a lower computational cost, as shown in figure 7.

Sampling algorithms are especially interesting for problems where it is difficult to obtain probability distribution information by other methods. If it were easy to use analytical methods, such as the derivative of the function, to learn key aspects of the function, sampling would not be necessary. Likewise, if it were a very simple function to sample because the function $f(x)$, which returns information about the function at a specific point x, is executed very quickly, it would not be necessary to use a subset of the population, the entire population would be sampled and thus a greater amount of information would be obtained with greater precision.

Therefore, sampling algorithms are used in those problems in which analyzing the set of

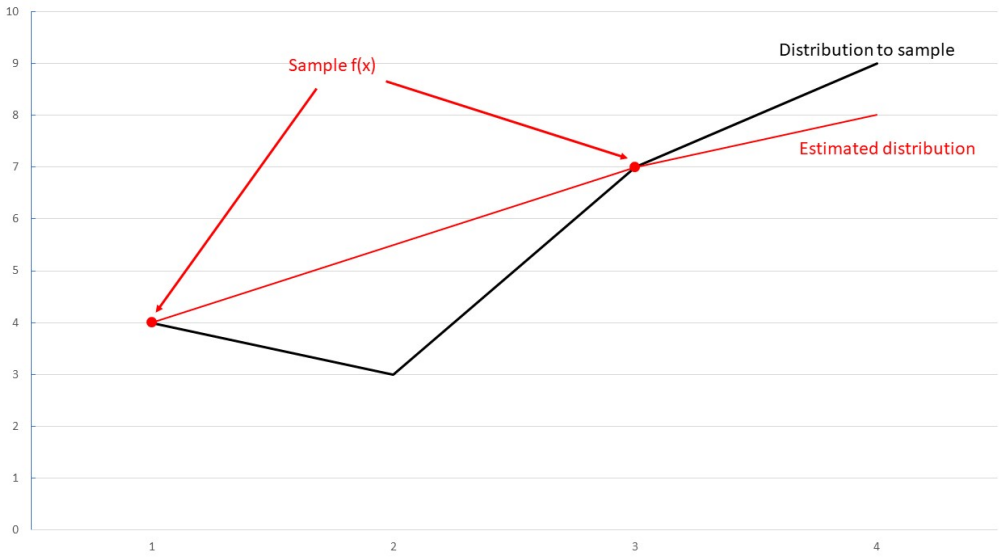

Figure 7: Scheme that shows how a distribution (black) can be estimated (red line) just getting some points (red circles) from it. The red line, the estimated line, is far from the real distribution because there are only two points sampled. Increasing the samples points, the estimated distribution will be closer to the real distribution to sample.

states is too complicated due to its size or the characteristics of the problem. Once again, large exponentially scaling state spaces reappear. This makes analyzing a small subset of data very useful but not the solution, because it has to be sampled individually. This legitimizes quantum techniques that allow for sampling over the superposition of all states.

Another fundamental characteristic of the problems on which sampling techniques are applied is the relationship between the elements that make up the population. It has been previously mentioned that the population must be large enough to be very slow to analyze all elements. In addition, there must be certain relationships between the elements so that when a subset of these is chosen, the properties and relationships that exist in the entire population are reflected with a certain probability in the reduced subset to be analyzed. Thus, general conclusions can be extrapolated to the whole problem.

Depending on the relationships between the data and the type of problem to be solved, one sampling algorithm or another can be applied within the wide range of possibilities for sampling a problem. Later, in chapter 2.2, classical sampling algorithms will be studied in detail. However, in order to have a global view, these algorithms can be classified as probability and non-probability sampling.

In probability sampling, the elements that make up the subset of the total population to be analyzed are chosen following a more or less random method. For this reason, any state of the population has a non-zero probability of being in the sample subset. This helps in choosing some confidence elements of the population that have general characteristics. Within this category are the methods of simple random sampling, stratified sampling, systematic sampling and clustering sampling.

Non probability sampling techniques use non-random sample selection. By eliminating the stochasticity of the process of sampling elements of the population, the selection is made by a criterion of convenience or explicit knowledge of the problem. This type of alternative selection makes the sample less representative of the population, but helps to avoid false data, outliers, or unbalanced samples.[47]

Using any of the techniques that belong to one of the above categories, a sampling error occurs. This error is due to the fact that a certain sample subset of a population may not contain all the characteristics of the population. In order to know a particular characteristic of the population, the sampling error is the difference between the estimated value of that parameter using the sample and the actual value of the parameter calculated using the entire population. To dampen this possible error, a confidence margin is used, which allows an estimate to be made within a certain confidence interval.

The error during sampling can be of different types and there are techniques to mitigate it depending on its nature and the algorithm used. This error causes the results obtained on the sample to differ from the real results that would be obtained on the population. This difference can be understood as bias because it gives information about the problem that is related and closely connected to the samples that have been chosen from the population. This bias is unavoidable during sampling, but it is not necessarily detrimental.

Biases can be voluntary or involuntary. Unintentional biases should be avoided because they reduce the generality of the results obtained and lead to erroneous results, precisely because of the ignorance of these biases. Voluntary biases, however, help to focus on a particular characteristic of the data set. This helps to avoid evaluating population data that lack relevance and to obtain more accurate information on certain parameters. For example, these voluntary biases are crucial in any machine learning algorithm because biases are the element used to generate a knowledge model of the problem. Having a knowledge model without biases means that the model is too general that it is not useful due to unspecific results.

Sampling algorithms are widely used to learn functions that are difficult to analyze with other methods. For example, for signal analysis, they allow to approximate the signal and reproduce it without having to evaluate it at all possible points. This is especially useful for continuous signals that need to be discretized. Another example is in the financial sector, the analysis of

the trend of an asset or the risk of a certain portfolio.

Another field of great application for sampling is machine learning. When running an intelligent algorithm that is trained on a set of input data, it is necessary to analyze these data to generate a set of hypotheses that explain them. In this hypothesis space, a sampling is made indicating how the data are being generalized, coverage of the hypothesis, and how complex they are, and length or number of hypothesis clauses. This is applicable to both supervised learning, for example, certain types of neural network, and unsupervised learning, for example, the k-means algorithm [48].

In the case of this work, sampling algorithms will be especially used to sample a large space of states, seeking to understand where a maximum or minimum of the probability distribution that defines a certain problem may be. In this way, sampling will serve as a first step to guide the search and make it over a much smaller state space. Thus, the better the sampling, the more accurate and faster the search will be.

1.3 Relevance of S&S

As explained above, the S&S algorithms share certain characteristics that allow a joint study of both techniques. This joint study can also be extended to the types of problems that can be solved with S&S algorithms. Some characteristics that have to be present in the problems are:

- **Defined initial state:** The initial state of the problem must be a defined state. It can be the initial state of the search or an initial point of the function to start sampling.
- **Transition function:** A definition of how one transits from one state to another. If it is a network, the connections and their probabilities would be defined; if it is a tree, the generation of successors, etc.
- **Evaluation function:** A function that can be applied in any state and provides information about the global or local state of the problem. For example, that calculates the cost of that state or of the path to reach it.
- **Well-defined Goal:** It must be clear how to solve the problem, whether the goal is to minimize the expenditure of a resource, maximize a certain reward, reach a particular state, etc.

Another common feature of S&S algorithms is the representation of the problem to be solved. Although there are different techniques [44], one of the most common is the Markov Decision Process (MDPs). MDPs are based on Markov chains determining that the probability of an

event occurring depends only on the immediately preceding event [49]. This type of causality hypothesis is very convenient because it ensures that the decision of the next state to be evaluated depends on the state of the system at the previous instant, facilitating the adaptation to changes and minimizing the possibility of getting stuck at a local level.

The MDPs define the problem as a tuple of (S, A, P, R) where:

- **S, States:** Set of actions called state space. It is the set that contains all possible values of the problem variables.
- **A, Actions:** Set of actions called action space. It is the set that contains all possible actions that the agent can perform in the state space.
- **P, Probability transition model:** It is the model that assigns a probability of each action that the agent performs in any state. The probability determines the new state after performing an action.
- **R, Rewards:** It is the expected return to the agent after performing an action in a state.

There are numerous variants of MDPs and they are specially designed to model the behavior of an agent in an environment, but they are suitable for modeling both sample and, especially, search. For sampling algorithms, MDPs are adapted in such a way that the agent has only one possible action to do, sampling.

Another important point to be emphasized in this paper is that S&S algorithms can act in concert. It is possible to iteratively combine first a sampling algorithm and then a search algorithm to reduce the search space in each iteration. It has been explained before that one of the difficulties of the problems in which S&S are applied is the size of the problem. Therefore, it seems logical to divide or reduce the size of the problem to make it manageable.

Especially in quantum computing, where hardware development is so limited, making an iterative algorithm that combines two techniques, search & sample, allows one to be quantum and the other classical. In this way, one technique will be able to process the state space in superposition and the other will be able to handle the large volume data set. This paradigm especially motivates the focus of this first part I of this thesis.

Some of the limitations of the S&S algorithms have already been succinctly described. Specifically addressing the complexity of these algorithms and the problems they solve, it is possible to define the complexity classes P and NP. These complexity classes serve to classify problems and algorithms into categories in which they share certain characteristics useful for finding a solution. The complexity determines whether a problem can be solved faster or slower, or

how much memory it will consume. The computational complexity classes are not invariant elements, but are open to constant revision and discussion [50].

Although the analysis of computational complexity is very extensive and there are numerous classes beyond P and NP, for the study of S&S algorithms, it is interesting to focus on these two.

P complexity class encompasses all those problems that can be solved in polynomial time. This means that a polynomial formula can be established that, given the number of variables in the problem, returns the execution time of the algorithm. A problem belonging to complexity class P is considered tractable because a classical computer can execute with relative simplicity a number of operations that grows polynomially. Therefore, problems belonging to P are said to have good scalability, since it is not expected that there are so many variables that the algorithm cannot be executed on a classical computer. An example of a problem belonging to P is addition; the number of operations is polynomial proportional to the number of variables to be added.

The NP complexity class is used to classify decision problems. The solution to these problems can be verified in polynomial time, but to find it a non-polynomial time is needed. Within this class, there are other classes such as NP-complete and also the NP-hard class [51]. The important thing about this class is that each time a new variable is added to the problem, the complexity or time needed to find the solution grows in a non-polynomial way. This implies that the scalability of this type of problem is poor and if the problem grows it is possible that it cannot be solved with a classical computer in a reasonable time [52].

The relationship between the complexity classes P and NP is somewhat unclear and is the subject of constant study. There are currently two currents among scientists, although one is assumed to be the most probable.

- **P = NP**: If classes P and NP are equal, it means that there are efficient algorithms for NP class problems that have not yet been discovered. This is the least accepted idea among computer scientists because it would mean that there is a way of doing algorithms that is completely unknown so far.
- **P $\neq$ NP**: If both classes are different, it means that there are problems belonging to NP that cannot be solved by a classical computer efficiently. This opens the door to the use of new computing paradigms that can work on this type of problem. In this thesis, hybrid quantum-classical computing algorithms are explored to tackle these NP problems.

Following this distinction between P and NP, it is easy to assume that P and NP are different, because although in theory they are not, right now, there are no algorithms to convert NP

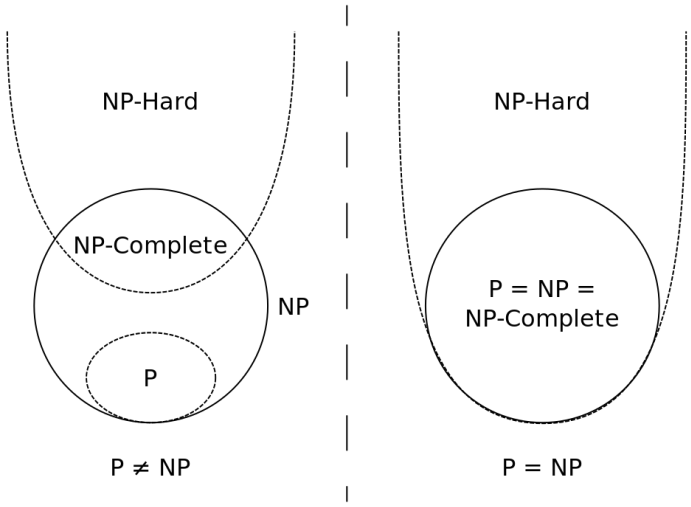

Figure 8: The complexity classes P and NP play a pivotal role in elucidating the computational complexity of existing algorithms. While the relationship between these classes remains elusive, prevailing evidence suggests that P and NP are not equivalent, as depicted in the left-hand scenario.

problems to P. Therefore, many of the problems to which S&S algorithms are applied are NP problems that cannot be solved classically faster. Some examples already mentioned are the TSP, the knapsack problem or the portfolio optimization problem. Figure 8 shows a scheme of these possible relationships between P and NP.

At this point, close to a dead end for classical computing, the option arises to think of new computing paradigms that approach the problem from another perspective. Quantum computing proposes to work with the entire state space in superposition. This new approach seeks the simultaneous modification of states by applying only one operation, as opposed to sequentially that would exist classically and that would require an operation for each state to be generated. This difference translates into the fact that quantum search is done exhaustively with the application of a single operator, while classically it is necessary for the operators to be much more precise in order not to have to do this exhaustive search.

While it is true that this superposition of states is a great advantage over classical computing, it also brings with it certain differences in information processing. While classically, the tricky thing is to generate states closer and closer to the goal state until it is reached, quantum computing is not about creating a superposition of states; the real tricky step is to increase the probability of those states that belong to the solution.

To be able to solve problems using quantum computing requires a new way of designing

algorithms, new ways of processing information, and designing specific operators. In this part I, it will be discussed how to design quantum S&S algorithms with advantages over the classical ones and how to apply them to industrial problems. For this purpose, a hybrid tool will be created that will combine the application of operators in quantum superposition with the processing of large volumes of classical data.

One of the main objectives of this work is to design and apply these new quantum computing techniques to create practical algorithms at an industrial level, including their implementations. This approach is part of a general trend in recent years that advocates to start creating algorithms that can be executed on quantum computers and not only theoretical proposals. In this chapter, how to apply S&S quantum algorithms to optimization problems will be discussed.

1.4 Optimization problems

There are many ways to define an optimization problem. In this thesis, a definition of optimization problems has been selected that is advantageous for the S&S algorithms. The problem definition consists of a successor generation function (probability matrix of transitions) and an evaluation function (reward function). In this way, any optimization problem can be defined as the probability of transitioning between states and the function that gives a value to that state or to the state of the world at that point. This definition of the optimization problem may certainly seem profitable for the interests of S&S algorithms, but the reality is that it is the definition that is common to most optimization problems at the industrial level.

The problems to be addressed in this thesis are the following:

- **Artificial intelligence:** In artificial intelligence, models with multiple variables that require a constant adjustment are used. This adjustment can be understood as an optimization of their weights to maximize a reward or reduce the distance between the state of the learning algorithm and the real world. [53]. Chapter 5 presents a study on how to apply a quantum S&S algorithm to the N-Queen problem. This problem is a common test-bed for search algorithms that optimize a certain learning function.
- **Space exploration:** In space exploration, it is common to collect large volumes of data. These data are often associated with problems that are studied using complex models. Applying a complex model to a large amount of data is costly in itself. However, in addition, these data often require some optimization. For example, in the detection of gravitational waves, it is necessary to make a parameter estimation that can be seen as fitting a certain function to the received wave by modifying its parameters. It will be discussed in more detail in chapter 6.

- **Quantum chemistry:** Quantum chemistry problems are especially interesting from the perspective of quantum computing. If, in addition, it is a problem that requires optimizing a particular configuration of variables, then quantum computation is more likely to work well. Chapter 7 addresses the protein folding problem, in which it is necessary to optimize the configuration of the protein to find the one with the lowest energy and thus avoid its denaturation inside the body.

The use cases discussed above share a common characteristic; all of their variables have a discrete domain. These types of problem are known as combinatorial optimization problems where having discrete variables, they tend to have a finite problem domain. Thus, the state space is large but bounded [54].

In the following sections, the algorithms currently applied to solve these problems and the quantum algorithms designed during this thesis will be detailed. In addition, it will be explained how to apply these quantum algorithms to the aforementioned use cases.

2

Classical Algorithms for S&S

Search and sampling algorithms have been extensively studied and evolved in classical computer science since the 1980s [30]. It is interesting to know how this evolution has been, to understand that the initial performance limitations have always existed and have been avoided in different ways as the theory evolved.

Whenever a new search or sampling algorithm or paradigm has emerged, it has evolved until a solution was found that faced an insurmountable performance limit. Once that limit has been reached, an attempt has been made to move forward by changing paradigm. For example, in heuristic search, different algorithms were created until stabilizing in the A* algorithm, which has remained the fastest optimal heuristic search algorithm for almost 50 years. In that time, other algorithms have emerged but with suboptimal search, other heuristics, etc. But the A* algorithm remains the most widely used in practice because it perfectly combines a fast and informative method.

The following is an overview of the main classical search & sample algorithms. After this review, the different limitations they currently have are analyzed.

2.1 Classical Search

Search algorithms are abstract techniques to find a state that meets certain conditions from a set. Depending on the characteristics of the problem, one algorithm or others can be used. The main types of search are shown in figure 9.

Due to this definition can be somewhat abstract and difficult to visualize, initial search algorithms were illustrated as trees. This metaphor serves both didactic and practical purposes

by being able to easily represent the evolution of the search algorithm.

In this way, the state in which the search starts can be seen as a node called root. This root node is evaluated to decide whether it is a node that represents a solution state. If not, it is expanded to obtain its successors. When a node is expanded, it is passed to a list of visited nodes. The successors go to a list of expanded nodes, not visited nodes, from which one is chosen to continue the search. Thus, the tree search is a process of visiting a node, expanding successors, and choosing a successor to repeat the process.

These search trees can have many shapes and characteristics. It may be that the same node can be reached from several paths, there may be loops between nodes, there may be nodes that have no children and are known as leaf nodes, and so on. What is important is that during the search, the algorithm is able to avoid two fundamental problems, getting stuck in a loop and redundant paths. To do this, many search algorithms check which nodes have already expanded in the list of expanded nodes and avoid entering them in the list of nodes to expand, the list of successor nodes.

These algorithms are classified according to their ability to achieve the concepts of optimality and completeness. Assuming that each expansion of node, successor generation, has a cost of 1.

- **Optimality:** Optimality means that the algorithm guarantees to have found the minimum cost solution.
- **Completeness:** The completeness of an algorithm guarantees that it is able to find the solution no matter which node of the state space it is in.

To review the most important search algorithms that exist and are relevant for the development of quantum search algorithms, a division will be made between informed and uninformed search algorithms. Within this category, the concepts of completeness and optimality will be analyzed.

Uninformed search algorithms are characterized by having no information about how close or far away the goal state is. This type of algorithm simply expands the nodes until one of them is found to be the goal. Returning to the tree scheme, the quality of the solution will be given by the distance from the goal node to the root, closer implies better quality. The cost of the algorithm will be given by the number of nodes needed to expand until the solution is found.

The main uninformed search algorithms are divided into sequential or random search and with different algorithms in each category.

- **Sequential:**

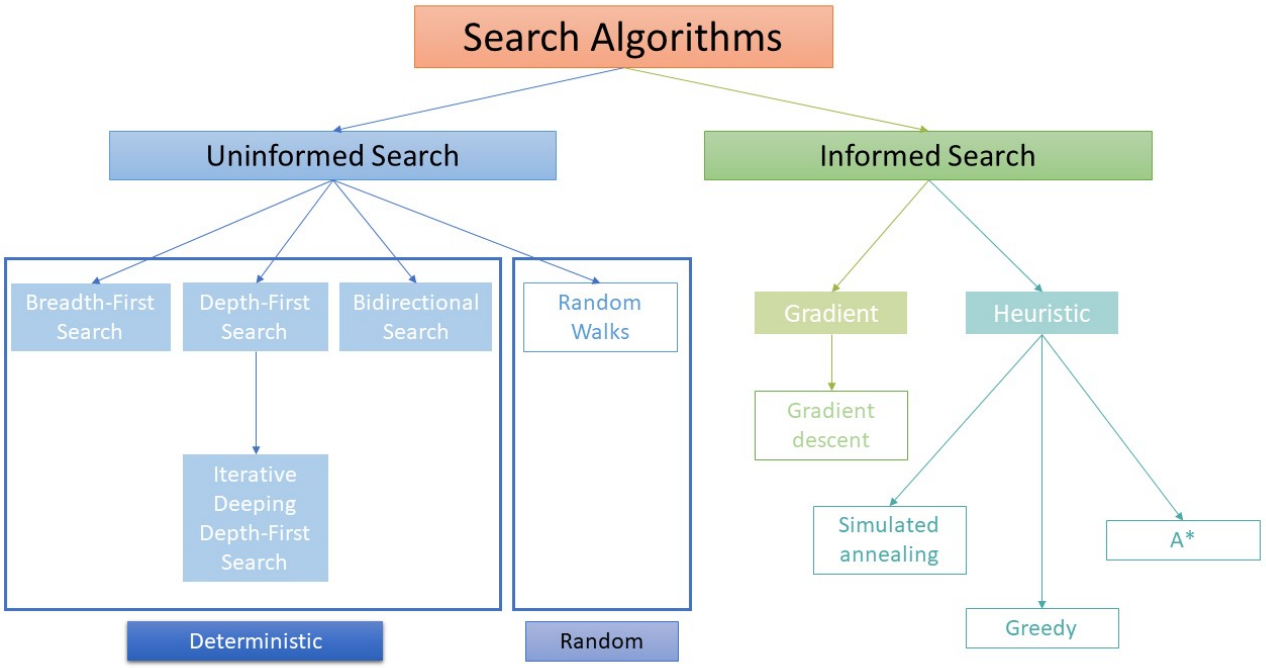

Figure 9: Scheme with the main search algorithms divided into categories attending to the type of search. Depending if the search uses any information of the problem to be guided, the search can be uninformed or informed. Inside of the uninformed category, there are deterministic searches or random algorithms. Among informed search algorithms, the information can be given using a gradient or an heuristic.

  - Breadth-First Search
  - Depth-first Search
  - Iterative deepening Depth-limited search
  - Bidirectional search

- **Random:**
  - Random walks

The breadth-first search was the first attempt to do a search in which, even if it did not have information on what state the goal was in, it might not have to expand all the nodes to find it. This type of search starts at the root node, then visits all expanded nodes from the root, and saves the successors of those nodes for the next round. In the next round, it expands all nodes at a distance of two from the root and saves their successors. In this way, it iteratively traverses

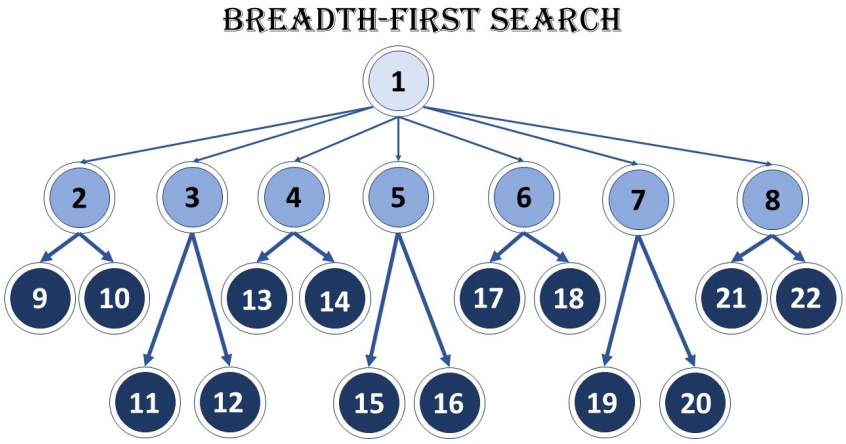

Figure 10: The breadth-first search algorithm systematically explores nodes at each level of a graph before descending to deeper levels. In this figure, nodes are organized into three depth levels, each represented by a distinct shade of blue, ranging from lighter to darker. Prior to traversing nodes at the third depth level, all nodes at the second level are exhaustively visited.

the entire breadth-first search tree, visiting all nodes at one level before jumping to the next. An example can be seen in Figure 10.

This type of search was initially proposed because it satisfies the criteria of completeness and optimality. By traversing all nodes at all levels of the tree, it is certain that at some point it will find the solution and it will be optimal. Optimality is ensured because it first visits all nodes of minimum cost, then all nodes of the next cost value, and so on.

Although this algorithm is perfect from the formal point of view because it meets two very important criteria, it always finds a solution and this solution is of the highest quality, in practice it is not a very useful algorithm. The reason for this lack of usefulness in practice is the number of nodes required to expand. It is a level-by-level exhaustive search algorithm, that is, until the solution is found, it will have expanded all the nodes [55].

The depth-first search follows the opposite paradigm to the breadth search. This search always expands the deepest node among the possible successors. Once a node that has no successors, leaf node, is reached, it returns to continue expanding the next deepest nodes. An example can be seen in Figure 11.

The search in depth was initially proposed at the same time as the search in breadth, but later forgotten. Its main problem was that it was neither a complete nor an optimal algorithm. In case of trying to find the solution in a very deep tree, the algorithm was lost by expanding nodes

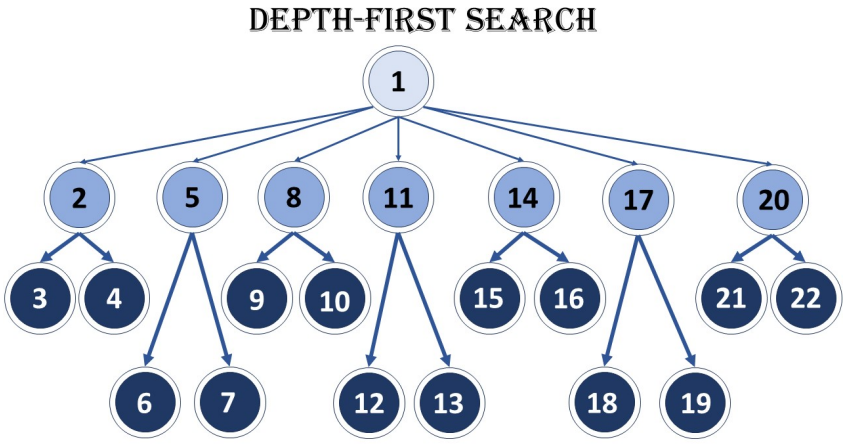

Figure 11: The depth-first search algorithm explores nodes along a single branch until reaching a leaf node, then moves to the next branch. Unlike breadth-first search, the depth level is not prioritized. In contrast to the approach depicted in figure 10, this algorithm focuses on swiftly reaching a solution rather than exhaustively searching all nodes.

even if the search was at shallow depth, which resulted in an unreliable technique. Moreover, even if a solution was found, it was not possible to know if it was at minimum depth, and it was necessary to continue the search almost as if it were in breadth until other solutions were discarded.

To solve the problem of the previous algorithm and try to take advantage of all its benefits, a modification known as iterative deepening depth-limited search was developed. The advantage of this new version of the algorithm was that it searched in depth up to a certain depth. Once that depth was reached. The algorithm went back to the beginning and started again from the root or the previously marked depth. The depth limit was getting bigger and bigger, deeper and deeper, until the solution was found. An example is shown in figure 12.

By having a bounded depth at which the algorithm stopped, the problem of infinite search was avoided. This allowed the algorithm to be complete and also, in many cases, to find the solution faster than the breadth-first search algorithm. It makes the iterative deepening depth-first search algorithm the preferred algorithm when an uninformed search algorithm had to be chosen. It is even possible to view this algorithm as a breadth-first search, if at each iteration one adds one to the limiting depth to search. In this case, the algorithm will perform a breadth-first search at each level. In contrast, if the maximum depth allowed is greater, it will find the searched node earlier or, in the worst case, with the same complexity as the breadth-first search, always faster or equal.

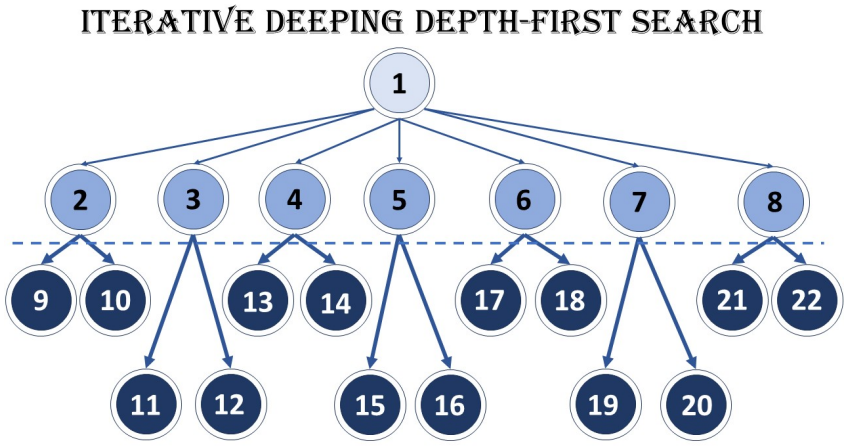

Figure 12: This scheme illustrates the order of node visits in the iterative depth-first search. Here, nodes within the same branch are explored until reaching a specified depth limit, denoted by the dashed line (set at 2 in this example). The distinctive feature of the iterative depth-first search, as opposed to previous algorithms, is its initial behavior resembling breadth-first search, as depicted in Figure 11. Once the limit is reached, however, the search becomes exhaustive, visiting all nodes less depth than the limit.

The bidirectional search executes two searches simultaneously. One from the initial state forward and one from the target state backward. In this way, both searches will meet at one point, and a path can be established between the initial state and the goal state.

This search is more specific because it requires that the problem have specific characteristics. The goal state must be known, and the problem must consist of finding the path to that goal state. In addition, this search algorithm, at each node visited, must not check if it is the goal state; it must check if that node has already been expanded by the other part of the search.

This algorithm has completeness and, moreover, can be optimal, provided that certain checks are made when both forward and backward searches meet. This bidirectional search algorithm plays with the boundaries of both search processes to try to minimize the number of states visited and reduce the complexity of the algorithm.

Unlike a sequential search, there are random search algorithms. The advantage of a randomized algorithm is that the successor generation time is reduced to a minimum. The stochasticity allows that, instead of processing which node is the best to visit, one is chosen randomly to continue the search. The drawback of this search is that it may repeat nodes and return low-quality solutions.

Random searches are a solution to problems where there is little information about the environment or where it is very complex and expensive to evaluate which type of search works

best. In addition, random search behavior is very easy to implement.

There are different random search algorithms. The most important for this work is the random walks algorithm. A random walk is a type of Markov process in which the algorithm explores the state space in a completely random manner. The idea of these algorithms is to make a search in the state space as fast as possible and without regard to the repetition of nodes or the efficiency of the search. The main advantage, in addition to speed and derived from stochasticity, is that a random walk never gets stuck at local minima or maxima and therefore, with some probability, makes a uniform state space coverage. The main drawback is that there is no information about the quality of the solution. It cannot be known whether the solution is optimal or how far it is from the optimal solution.

Random paths have different applications. For example, they are very useful to simulate the Brownian motion of particles or to perform fast searches to give a first solution that can be later refined, as in link prediction or semi-supervised learning [56]. Although one of their main applications, and the reason why they are studied in this work, is because they are useful as a base technique for other more complex informed search algorithms.

If during the search, the expansion of nodes is guided by some knowledge of the problem environment, it is considered a guided search. The main advantage of this type of search is that it generates nodes more intelligently, resulting in fewer nodes being expanded until the solution is reached. In addition, it can allow knowing the quality of the solution, knowing if it is optimal or the distance to the optimal solution. In this work, gradient search and heuristic search are going to be analyzed.

Informed search is currently the most widely used for solving search, optimization and artificial intelligence problems. This type of search has proven to be useful because, by selecting nodes with certain criteria, it considerably reduces the search space explored. Depending on the information that the algorithm uses to guide the search, a distinction is made between searches with domain-independent and domain-dependent information.

Searches using domain-independent information do not need to know specific information about the problem, only about the search process itself. For example, heuristic searches based on landmarks [57]. These domain-independent searches are useful if little domain information is known, but tend to be slow and explore a larger set of state space. In contrast, domain-dependent searches, although they require fairly precise knowledge of the problem being solved, allow very precise searches, which do not explore as many nodes of the state space and, therefore, are faster.

Gradient descent search analyzes the value differences, from a value function, between the different nodes of the search space to always advance to the point of minimum energy. In

this case, value can be understood as an evaluation function or cost function constructed with specific knowledge of the problem. For example, in the case of neural networks, the gradient is derived from the differences between the predictions and the actual data calculated by a function such as the least squares error.

Gradient descent follows minimization schemes with a convex distribution and one absolute minimum and, in some cases, several local minima. The goal of the search is to find as quickly as possible the node that represents the absolute minimum of the function without getting stuck in local minima. To do this, it performs an analysis of the slope of the function at each node it analyzes that lets it know the direction in which it has to continue the search.

Instead of using a gradient, a function that returns an estimate can be used to guide the search. This scheme is known as heuristic search. The heuristic gives information to the algorithm on how it should guide the search based on the distance to the goal or the time of the search. There are different types of heuristic search and numerous algorithms. This search, by expanding a node, calculates the value of the heuristic for each of its successors. Then, taking into account the heuristic value of each node, it chooses the one of greatest interest following a certain criterion.

The heuristic is a function h(x) that returns a value, where x is the node to be evaluated. To construct such a function h, the constraints of a problem are relaxed and simplified. The key is to make h as informative as possible, but also fast to compute. For example, if the search problem is to find the fastest path to the center of the maze, one possible heuristic would be the Euclidean distance, which is a simplification because it ignores the walls of the maze.

Greedy search is the simplest type of heuristic search available. In this case, the algorithm will always choose the node with the best heuristic value. In this way, the algorithm will blindly rely on the ability of the heuristic to approach the solution. This strategy has a very clear limitation, since the heuristic is a simplification of the problem, there will be times when the greedy search will fall into local minima and will not be able to get out.

The simulated annealing algorithm uses heuristics to tell the algorithm when to perform an exploratory search and when to perform a focused search within the explored space. This algorithm, based on metallurgical processes, uses a random walk algorithm. Initially, it imposes a high temperature that makes the selection of the next node to visit completely random. However, as the number of iterations of the algorithm progresses, the temperature decreases and the selection of nodes is less determined by chance and more by a decrease in the energy of the nodes. This algorithm makes it possible to combine a broad initial exploration with an accurate search after a given number of iterations.

The most important heuristic search algorithm is A*. This algorithm is similar to the greedy

search, but instead of selecting the node with the highest heuristic value, it takes into account the cost of the previous nodes expanded so far. Thus, not h(x) but $f(x)$ is used to choose which node to expand. $f(x)$ is equal to the sum of h(x), the heuristic value of the node, and g(x), the actual non-heuristic cost of having reached that node. A* solves the problem of getting stuck in local minima and, in addition, ensures that the search is optimal [58].

In this section, two different ways of guiding an informed search have been discussed, one based on gradient and the other based on heuristics. This analysis, which classically has some relevance, quantumly becomes of vital importance. As will be seen in the following, there are hybrid classical-quantum algorithms that use the classical gradient-guided search to optimize quantum circuits. It has been seen that this search suffers from a problem called barren plateaus (BP), which makes it difficult to find the absolute minimum. A possible solution may be to use heuristic-guided and non-gradient guided searches to avoid the BP phenomenon.

As it has been explained, the classical search, both informed and uninformed, is a broad topic with numerous variants, a small part of which has been analyzed.

While it is true that these search algorithms have numerous applications, it is also necessary to understand that they have clear limitations. These limitations are so strong that they result in a very limited scalability of the algorithms. As a result, search algorithms make slow progress in solving increasingly complex problems and are very limited in solving optimization problems in industrial environments. In 2.3, the factors that limit these algorithms, their consequences, and how they are tried to be solved will be analyzed.

2.2 Classical Sampling

Classical sampling algorithms are techniques that allow to know a probability distribution P of a population X just analyzing a subset x of this population called sample. The objective of the sampling algorithm is to obtain an estimated probability distribution $\hat{\mathrm{P}}$, as close as possible to the real P, by analyzing the smallest sample x possible.

During the sampling process, a sampling error occurs. This error is the difference between the data estimated by the distribution obtained from the previous samples and the real value of that sample. This error becomes smaller as more samples are taken. A sampling algorithm must always maintain the balance between analyzing a few samples and maintaining a small sampling error.

In general, sampling can be classified as probabilistic and non-probabilistic. The main difference lies in whether the samples of the population are taken completely randomly following a certain criterion, probabilistic sampling, or whether samples are chosen from a limited subset

of the entire population following a specific criterion or bias, non-probabilistic sampling.

Probability sampling is more complex and slower to perform because there is no a priori information about the distribution being sampled, and it is not simplified in any way. The advantage is that the sampling process does not introduce any bias, so that the function and parameters learned from this type of sampling are much more general. This work focus on probabilistic sampling techniques, specifically on Monte Carlo methods and derivatives.

The Monte Carlo sampling method can be defined as a probabilistic sampling method used to approximate mathematical functions. This method is integrated within a larger family of algorithms that allows to know and perform certain estimations on physical experiments using random sampling [59]. The Monte Carlo method is based on sampling a function completely randomly until there is a sufficient number of samples to know this function in sufficient detail. Thus, the Monte Carlo method calculate an approximate estimation of the real probability distribution.

This estimation obtained by the Monte Carlo method will have a certain error that will depend on the number of samples used during sampling. This error decreases as $\frac{1}{\sqrt{N}}$, where N is the number of samples taken from the function.

The Monte Carlo method is very useful for approximating certain mathematical functions, but if the objective is to approximate the distribution of a particular problem about which there is some a priori knowledge, it is possible to use more accurate methods based on Monte Carlo sampling. One such method is Metropolis-Hastings.

The Metropolis-Hastings algorithm is a method that combines the above elements of Markov and Monte Carlo processes to construct a Markov Chain Monte Carlo (MCMC). This method uses domain-specific knowledge to guide the sampling method between the proximate values. This type of sampling can also be understood as a search guided by a heuristic, generally based on energy minimization.

The Metropolis-Hastings method uses random paths to decide which node to evaluate or expand next. Once the next node has been randomly selected, it compares the value of the evaluation function of both nodes. If the problem is a minimization problem and the energy of the new node is lower, the change is accepted, and the algorithm moves to that new node. If, on the other hand, the energy of the new node is higher, the change is initially rejected. However, the energy difference between the two nodes is analyzed and, with a probability proportional to the energy difference between the two nodes, the new node is accepted. This probability of accepting a transition to a node that is not better than the previous one is essential to prevent the algorithm from stagnating in local minima. These steps are shown in figure 13. The Metropolis-Hastings algorithm is repeated until a converge criterion is found. The criterion can

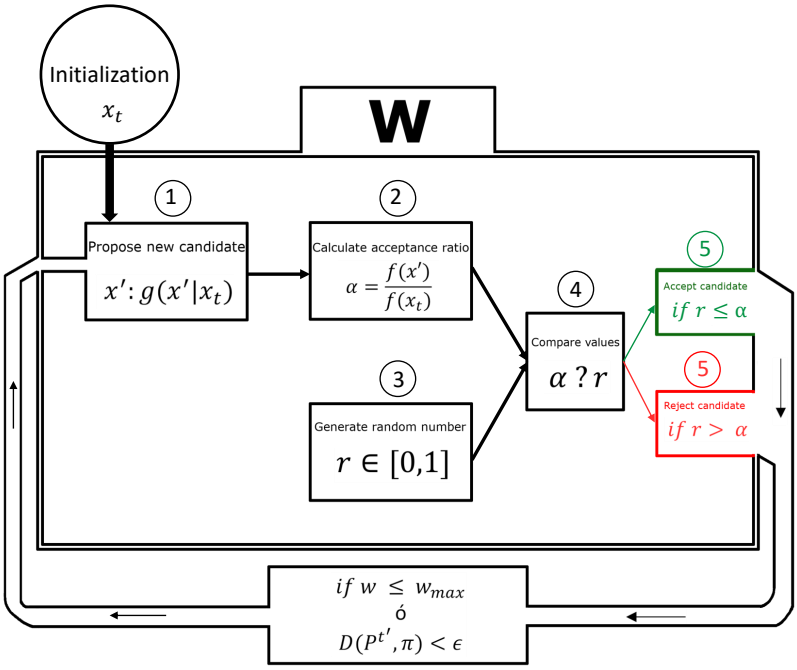

Figure 13: This scheme shows the Metropolis-Hastings algorithm process to propose a new candidate and reject or accept it. This process is divided in 5 steps. Step 1 consists in a random candidate proposal. In step 2, an acceptance ratio is calculated using the evaluation function for the actual state and the candidate selected in step 1. In step 3, a random value between 0 and 1 is calculated. In step 4, both values, acceptance ratio and random value, are compared to decide if the new value is accepted or not. In step 5, the candidate is rejected or accepted. This iterative process is repeated until a maximum number of steps is reached or the distance between the real and the inferred distribution is below an epsilon value.

be that a maximum number of Ws has already been executed or the distance D between the calculated distribution and π is lower than an epsilon value, ϵ.

The acceptance ratio of the change is defined as

$$A_{ij} = \min\left(1, e^{-\beta(C_j - C_i)}\right), \tag{1}$$

In this equation, i is the current state and j is the candidate state. C_i and C_j are the costs of these new states and β is a parameter representing the inverse of the temperature. With more temperature, the algorithm performs a wider exploration, with lower temperature, the algorithm focuses on finding a local minimum.

The Metropolis-Hastings algorithm is particularly useful in bayesian contexts where one seeks to approximate the posterior distribution. In these cases, the functional form of the posterior distribution is difficult to compute analytically. By sampling in parameter space, a representation of the posterior distribution is obtained that can be used to estimate credible intervals, means, and other statistics of interest. The likelihood of each new sample is decided with a function that must be proportional to the posterior that from where sample are taken.

As can be seen, the Metropolis-Hastings algorithm aims to sample a function but uses methods very similar to those of the informed search. In fact, it is not unreasonable to think that a pseudo-random search can be performed using Metropolis-Hastings so that each step of the algorithm maintains an appropriate balance between exploring new paths and exploiting paths already visited that are close to the solution.

2.3 Classical limitations for S&S

As seen in the previous two sections, both classical search & sample methods have been extensively studied and developed. Moreover, they have numerous industrial applications in which they solve problems that, although small, are totally impossible to solve in any other way. However, more complex problems that can only be solved by high-performance computing (HPC) require algorithms with good scalability that are capable of solving larger and larger problems. This type of problems, usually combinatorial optimization problems, have the particularity that their complexity increases exponentially.

If problem complexity scales exponentially but problem analysis capabilities scale linearly, a limit is soon reached [60]. Increasing the size of the problem by a single unit takes the problem from being solvable by HPC in days or weeks to being unsolvable because its execution time increases to decades or centuries [61]. One of the main reasons for this poor scalability in search & sample algorithms is the phenomenon known as the curse of dimensionality.

This phenomenon occurs when the dimensionality of the data grows very fast, causing the volume of the data to grow as well. As a result, the data become scattered and difficult to cluster [31]. The curse of dimensionality causes that in order to find a solution in the data, it is necessary to evaluate many more dimensions as the problem grows because the state space becomes sparse. This problem is difficult to solve classically and, for that reason, this paper proposes a quantum solution for the search & sample algorithms.

3

Quantum Algorithms for S&S

Quantum search & sample algorithms (QS&S) represent a paradigm shift in the approach to solving problems involving some form of quantum algorithms for search or sampling. It is quite intuitive and easy to apply it to problems where it is necessary to analyze the entire search space and, therefore, use exhaustive algorithms. However, QS&S algorithms are also useful if to find the solution it is necessary to explore an important part of the state space and this is spatially very large.

As discussed above, the curse of dimensionality affects classical algorithms because they must perform an exponentially larger number of operations even though the dimension of the data increases to a lesser extent. Faced with this problem, QS&S algorithms are interesting because their complexity is not so dependent on the data dimension, as will be seen in some of the algorithms explained below. However, this is only an intuition followed by the community that has not yet been proven. In fact, there are some authors that call for caution with this type of algorithms and problems, saying that it is necessary to find better advantages for quantum computing to be really useful [62].

In quantum algorithms, the winning bet seems to be the reduction in the number of operations needed to find a solution. This lower number of operations means that fewer quantum gates than classical ones have to be applied. If this statement proves to be correct, it will ultimately result in fewer operations, which, due to the physical foundations of quantum computing, will also translate into lower energy costs. This saving in the energy spent on executing an algorithm to solve an optimization problem, for example, is not trivial, since classical computers, specifically HPC centers, often have an energy consumption that is unbearable for computing centers [63].

Problems solved by QS&S algorithms have been of great interest since the inception of the concept of quantum algorithms. It was considered that in search, sampling and optimization

problems, QC would have a great impact. Therefore, a part of the academic effort was devoted to this topic, giving rise to numerous techniques and algorithms. In this section, main ones are going to be analyzed. First, the Grover operator will be analyzed because it is a technique commonly used by other algorithms. Based on Grover, it is possible to study quantum walks, and their implementation in the quantum Metropolis-Hastings algorithm, and quantum backtracking algorithms. Also, algorithms based on quantum annealing will be analyzed because of their wide use and easy implementation in adiabatic hardware. Finally, an analysis will be made of how these algorithms can be adapted to the hybrid classical-quantum paradigm, with special interest in the quantum Metropolis-Hastings (QM-H) algorithm, which will have special relevance in this work.

It should be noted that these QS&S algorithms have a wide range of applications for both theoretical and real problems. Although the problems currently being tested are restricted to an academic rather than an industrial setting due to the limitations of quantum HW, it is expected that in the short-medium term, QS&S applications will start to be noticed at an industrial level. In this work, practical applications of QS&S algorithms are going to be shown as use cases with proofs of concept using real data of the problem they are solving, in order to bring a little closer the demonstration of the practical utility of these algorithms in these problems.

3.1 Grover Operator

The Grover operator did not emerge as such, but was proposed as an algorithm by Lov K. Grover [13]. After being studied in depth by the scientific community [37], it was revealed as a basic technique for other algorithms beyond the initial applications intended by its creator.

The importance of Grover's algorithm/operator lies in the fact that the fundamental operation of classical computers is to store and retrieve information, i.e., to search and sort information in large volumes of data. Search operation has been isolated and studied since the first computers [29]. For example, one of the natural solutions to find information faster are indexes. Classical computers employ all kinds of indexes that allow searches in logarithmic complexities [64]. However, it is impossible to create indexes for all data sets because it would make the index search even slower than the search itself. Thus, the only solution is to create a reduced set of indexes for certain datasets and perform most of the searches exhaustively.

This exhaustive search, commonly called "brute force search", consists of making a visit to every possible state in the state space, which makes it a really expensive operation. The cost of the algorithm is defined as the number of queries or times that an element is evaluated. If there are N elements, the cost will be N. Grover realized that it was possible to perform a similar operation but with a cost of $\sqrt{N}$ using a quantum algorithm.

To see it with a practical and very simplified example to clarify the concept, if in a library there are 100 million books without any order, and a specific book is required, in the worst case, it is necessary to read the title of 100 million books. This task is practically unfeasible for a single person. Using Grover's algorithm, it would be necessary to make an effort equivalent to reading only 10,000 titles to find the right one, which, although tedious, is perfectly feasible for one person. It is said that the quantum algorithm makes an effort equivalent to reading 10,000 titles because there is no exact equivalence between reading a title classically and having it in quantum superposition, but the analogy of complexity can be established.

To construct his algorithm, Grover uses an oracle $f(x)$ given $x \in \{0, 1, ..., N-1\}$. This oracle evaluates each x, each possible state of the system (a title of a book in the previous example). Then, the oracle can take two values, $f(x) = 1$ if the x value is the desired value, $f(x) = 0$ in any other case. If this oracle is applied to a superposition of all possible values x defined in Eq. 2. The superposition is created, applying a Hadamard gate to each qubit representing the state that initially is in the state $|\cdot\rangle$. Being $N = 2^n$, the oracle will only return 1 when it matches the correct value, and the task will end.

$$H^{\otimes n} |\cdot\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \,. \tag{2}$$

It seems that the algorithm is complete, and it will return 1 in the correct state. Nevertheless, quantum computing is not so simple. Although it is true that after executing the oracle once, the probability of the correct state will go up, the difference between the probability of the correct state and the rest of the states will be so small that when measuring the circuit, it will not differentiate which state has the highest probability. This is a common occurrence in quantum computing in many algorithms, and, therefore, it is necessary to repeat a certain set of gates several times until the probability of the correct state is sufficiently large with respect to the others.

In the case of Grover's algorithm, this increase in the probability of the marked state is done by repeating two rotation operators. These two rotations play with the phase of the states, making an inversion about the mean probability. The first rotation marks the state in which $f(x)$ is 1 by associating with it a phase opposite to that of the other states. The second rotation, known as the diffusion operator, performs a rotation on the initial state, converting the phase difference between the states into probability amplitudes. In this way, the marked state will have a higher probability of being measured.

The first rotation in Eq. 3, which is identified with the oracle, is responsible for marking the state sought.

$$R\left|x\right\rangle = (-1)^{f(x)}\left|x\right\rangle. \tag{3}$$

The result of applying the operator R, oracle, to the superposition of states in Eq. 2 is a 1 in all states, except in the searched state that is -1. Then, the second rotation, the diffusion operator, has to be applied. The rotation operator is

$$U_D := H^{\otimes n}(1 - 2\left|0\right\rangle\left\langle 0\right|)H^{\otimes n}. \tag{4}$$

This diffusion operator U_D takes all possible states in superposition, applying the Hadamard gate $H^{\otimes n}$ to the reflection on the average value that was previously marked with a -1 ($(1 - 2\left|\cdot\right\rangle\left\langle 0\right|)$). Using a simplified example [65], given a state $|\psi\rangle$ with 4 possible elements, each of them with a probability of $\frac{1}{4}$ and a coefficient, amplitude of probability, of $\frac{1}{2}$.

$$|\psi\rangle = [\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]. \tag{5}$$

If the marked element is in the second position, after applying the oracle in Eq. 3, the resulting state is as follows:

$$|\psi\rangle = [\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}]. \tag{6}$$

The mean of this state $|\psi\rangle$ is $\frac{1}{4}$. Applying equations 3 and 4 multiple times to ψ following the Eq. 4, the resulting state is:

$$|\psi\rangle = [0, 1, 0, 0]. \tag{7}$$

The 1 is in the second position that corresponds to the searched element. It is an explanatory example, very simplified, but it helps to understand the mechanism used by Grover to its algorithm.

These reflection and diffusion operators went far beyond Grover's algorithm and have become really important in the quantum search algorithm community. Beyond their application in non-indexed key search problems, they have been applied to all those algorithms that needed to amplify the amplitude of a particular state [66].

3.2 Quantum walks

The classical search algorithms explained in 2.1 had a common problem, the need to search over almost all, if not all, nodes in the state space in order to find the solution state. One of the algorithms explained, the random walks, proposed to make a very fast but low quality exploration, so that they would explore with a certain probability a region of the search space large enough to find a solution of approximate quality to the optimal solution.

The main problem with random walks was that, although fast in execution, it was very expensive for them to explore a significant part of the search space. Moreover, this non-repetitive exploration was not guaranteed due to the stochastic nature of the algorithm. For that reason, it was necessary for them to execute a high number of repetitions to ensure the correct exploration of the state space.

In the quantum case, the state space can be represented as a superposition of states, applying each operation over the entire search space at once. This solves the problem of the exploration limitations of random walks by switching to their quantum counterparts, called quantum walks. In the case of quantum walks, the difficulty does not lie in exploring all possible states, but it becomes complicated to differentiate which state should be chosen because the operations are applied over the entire superposition, not to individual states.

Any type of walk is easy to imagine, thinking of a graph G with vertices (states, S) and edges (transitions between states, T). This graph is traversed by an agent that makes a path by randomly transiting S states. From a state s to another state s', it will transit with the probability defined in the edge t. The agent can take a series of random W steps until it reaches a state that will be considered final.

Ambainis' initial proposal for quantum walks [36], taking advantage of the Grover operator, demonstrated a quadratic advantage over classical random walks. Another quantum walks model, introduced by Szegedy [67], is based on graphs and represents the state space as a bipartite graph, as explained in [68],

Szegedy defines the bipartite graph as two Hilbert spaces $\mathcal{H} \otimes \mathcal{H}$, where each $\mathcal{H}$ represents the state space X. These are two identical copies of the original space. Then, two operators to transit around the graph are defined:

Update operator

$$U\left|j\right\rangle\left|\cdot\right\rangle := \left|j\right\rangle \sum_{i \in X} \sqrt{W_{ji}}\left|i\right\rangle = \left|j\right\rangle\left|p_j\right\rangle, \tag{8}$$

and

$$V\left|\cdot\right\rangle\left|i\right\rangle := \sum_{j\in X}\sqrt{W_{ij}}\left|j\right\rangle\left|i\right\rangle = \left|p_i\right\rangle\left|i\right\rangle. \tag{9}$$

The matrix W_{ij} can be seen as the weights of the edges T in the graph, which represents the probability transition from $|i\rangle$ to $|j\rangle$. U and V are related as:

$$SU = VS, \tag{10}$$

where S is the swap operation, to move from one $\mathcal{H}$ to the other. Then, it is possible to define the subspaces A and B.

$$A := span\{|x\rangle\left|\cdot\right\rangle : x \in X\}, \tag{11}$$

$$B := U^{\dagger}VSA = U^{\dagger}SUA. \tag{12}$$

The projector applied to both subspaces:

$$\Pi_A := (1 \otimes |0\rangle\langle 0|), \tag{13}$$

$$\Pi_B := U^{\dagger}VS\Pi_A SV^{\dagger}U = U^{\dagger}SU\Pi_A U^{\dagger}SU. \tag{14}$$

Using rotations similar to the Grover proposed, the quantum walk operator defined by Szegedy is defined as:

$$W = R_B R_A = U^{\dagger}SUR_A U^{\dagger}SUR_A. \tag{15}$$

The quantum walk just explained is of the time-discrete quantum walk type. This allows to execute each step of the quantum walk independently and to adjust it to the reasoning model of the Markov agents.

As seen above, classical random walks were slow due to their random exploration that could repeat multiple states, something that does not happen with their quantum counterpart. Quantum walks are especially interesting in those problems where a virtually exhaustive search of

the state space has to be performed. Just as Grover talked about an exhaustive search in a phone book or in the library example, if a quantum walk has to be run on a problem that needs almost all the nodes of a state space, it will find an advantage in having the state space in superposition.

Yet, practical implementation limitations hinder the realization of theoretical quantum walk proposals, particularly Szegedy's idea of representing the entire state space in a bipartite graph. Consequently, certain operators within W must be modified to achieve a more efficient representation of the state space in the quantum walk.

Just as classical search serves as a basis for other algorithms, quantum search based on quantum walks is used on numerous occasions, both to accelerate classical algorithms in hybrid schemes and to build larger quantum algorithms. In [69] is explained how to search for marked elements in a Markov chain using the quantum walks proposed by Ambainis and Szegedy. In [70], a quantum search based on quantum walks is used to accelerate the process of exploiting a Reinforcement Learning algorithm.

As seen above, the difference between search & sample algorithms in quantum computing is not as clear as it is classically. The same is true for quantum walks. If the previous paragraph explained quantum walks applied to search, they can also be applied to sampling processes. The best example in this line is the work of Lemieux et al. [71] who modified the Szegedy operator by replacing the bipartite graph by a coin and converted the W operator of the quantum walk into an operator to construct a quantum Metropolis-Hastings algorithm, which is going to be explained in detail in the next section.

3.3 Quantum Metropolis-Hastings

In the work of Lemieux et al., they take the Szegedy quantum walk operator W and replace the bipartite graph representation with a coin that is in a superposition of 0 and 1. This coin is entagled with the state space, so that changing the probability of the coin also changes the probability of the states. This new quantum proposal of the M-H algorithm has three registers:

- $|\cdot\rangle_S$ is the register containing all the qubits needed to represent the entire state space.
- $|\cdot\rangle_M$ is the register containing the move qubits.
- $|\cdot\rangle_C$ is the register representing the coin, just one qubit.

Using these registers, the proposed operator is as follows:

$$U_W = RV^\dagger B^\dagger F B V. \tag{16}$$

The operators are:

- Move preparation operator V: It creates the superposition in the register $|\cdot\rangle_M$.
- Coin operator B: Rotates the coin according to the criteria defined in Eq. 1, similar to the classical case. It loads the coin with a certain probability and prepares it to the operator F.
- Coin flip operator F: It flips the states $|\cdot\rangle_S$ according to the probabilities in the coin, $|\cdot\rangle_C$.
- Reflection operator R: It is a reflection in the registers $|\cdot\rangle_S |\cdot\rangle_C$, derived from the Grover reflection operator. It is responsible for increasing the probability of states marked by the coin register $|\cdot\rangle_C$.

This operator U_W, in combination with the Markov models explained in chapter 1, is used to construct the quantum version of a Metropolis-Hastings algorithm. Similar to the graph representation, the agent that traverses the graph G can be modeled as a Markov model in which the decision to move to one or another of the possible states depends only on the state in which the agent is. The probability transition of the agent from one state to the next is codified in the rotation of the coin.

It is important to note that the operator proposed by Szegedy W and the one proposed by Lemieux et al. U_W are not equivalent. The new operator U_W is a very smart implementation because it avoids the need to uncompute the move register $|\cdot\rangle_M$ and the coin register $|\cdot\rangle_C$, which is a very costly problem when the new move proposal is rejected.

This initial proposal of a quantum Metropolis Hastings was focused on a specific problem, a one dimensional Ising model. This problem can be used as a simplification of other more general optimization problems. However, as demonstrated later, and as will be shown in the use cases, this version of QM-H can be adapted in different ways to solve more complex optimization, search & sample problems, such as protein folding [16], the N-Queen problem [72], parameter estimation in gravitational waves [2], etc.

The main advantage of this QM-H algorithm is the possibility of using it in combination with a heuristic function. This makes it possible to apply the algorithm to almost any optimization problem. As seen above, in cases of informed search, if one has some information about the problem domain it is possible to guide the search to speed it up. In the case of quantum algorithms, if the heuristic can be evaluated over all possible states, it allows the algorithm to have much more information about the problem with fewer operations.

The quantum M-H algorithm can be used for search problems because the agent always looks for nodes that minimize energy, except for some random movement with increasing energy to avoid local minima. In this way, the problem can be encoded in such a way that the algorithm searches for the absolute minimum of an energy function that can represent a minimum number of kilometers traveled, minimum budget spent or minimum amount of fuel consumed.

The key to the search performance of the quantum Metropolis-Hastings is its ability to sample the state space with high accuracy faster than the classical algorithm. This sampling is based on knowing perfectly the probability of acceptance of a change in the studied distribution, something that inherently adjusts the quantum algorithm to the problem it is solving.

3.4 Other quantum S&S proposals

In addition to algorithms based on quantum walks, there are other quantum computing proposals for S&S algorithms. Understanding the limitations of some of these proposals, strengthens the capabilities of quantum walk-based algorithms.

One of this proposal is quantum backtracking. Taking the example of an agent moving through a maze, or the problem of loading packages into a truck, a different type of search called backtracking is useful. The key point in backtracking algorithms is that they perform an exhaustive search, but in an intelligent way, avoiding exploring options that are already known not to lead to a valid solution. This helps improve efficiency compared to brute force approaches.

In this case, Montanaro [73] proposed to apply the techniques of quantum walks to speed up backtracking algorithms. The main function of a backtracking algorithm is that it represents the state space as if it were a tree. This algorithm expands and explores branches with different states until a solution state is found. The difference with other algorithms is the ability to evaluate the nodes in such a way that it can know if all the successors, all the branches, generated by that node are going to approach the solution or not. In case it has found an unpromising branch to find the solution, it returns, backtracking, to the root of that branch and takes a different one to continue the search as shown in figure 14.

The major disadvantage of quantum backtracking algorithms is intrinsic to their advantage. These algorithms inherit the state space representation limitations explained above for Szegedy quantum walks. They even add additional operators that, while making the advantage greater, also add a high complexity to bring them to a real implementation, either in simulator or hardware, that can solve a practical problem. The limitation of quantum backtracking remains the same as that of many quantum algorithms; it is still difficult to prove its advantage because the very small quantum hardware resources make its implementation impossible until several

```
procedure EXPLORE(node n)
    if REJECT(n) then return
    if COMPLETE(n) then
        OUTPUT(n)
    for n_i : CHILDREN(n) do EXPLORE(n_i)
```

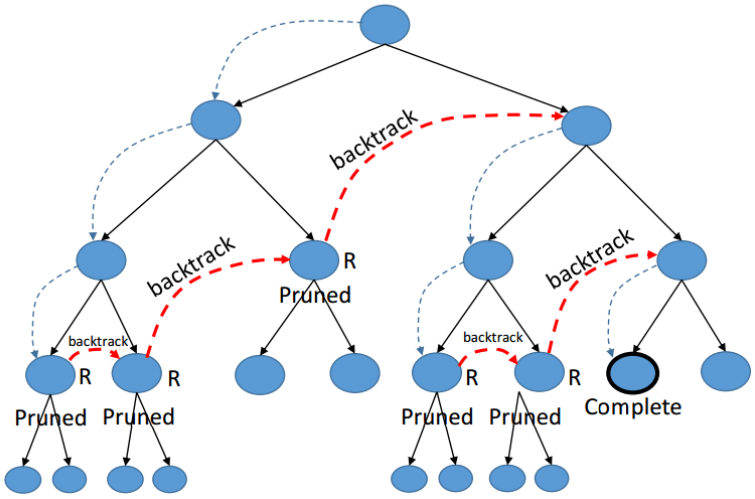

Figure 14: Scheme that shows how a backtracking algorithm works. This algorithm visits the nodes following a depth-first-search. If it detects that the solution is not in the explored branch, it prunes the branch and backtracks to a different branch. Extracted from [74].

years from now.

A different paradigm from quantum walks for implementing QS&S algorithms is quantum annealing. This proposal, which is almost at the same time an algorithm and device, since the hardware where it is executed is only dedicated to execute this type of algorithms, is inspired by the classical technique of simulated annealing. These algorithms start with an easily achievable quantum configuration and then, by gradually cooling the system, seek to "anneal" the optimal solution. The main idea is to configure the system in a quantum state according to the initial conditions of a problem, and then let the system evolve until it reaches a minimum energy configuration that corresponds to the solution of the problem [75].

To encode problems for quantum annealing, various methods can be employed. One of the most common approaches is to formulate them as unconstrained quadratic binary optimization (QUBO) problems. Such representations enable a binary encoding of the variables and constraints of the problem. In this way, the algorithm effectively solves a system of equations

where it needs to maximize or minimize certain values while adhering to specified constraints [76].

QUBO binary encoding can be quite rigid when it comes to representing complex structures that may be necessary in an optimization problem. Therefore, it has been relegated to a method of solving very specific problems. Nevertheless, it remains an alternative to consider for addressing certain QS&S problems [77]. Scalability is another limitation, closely related to the lack of a formal demonstration of its advantage over classical algorithms in large-scale problems. Although quantum annealing tackles problems of great complexity for classical algorithms, providing an advantage, the scalability of adiabatic quantum hardware devices has proven to be insufficient [78]. This lack of scalability eliminates the advantage for larger problems, leading to the acknowledgment of quantum annealing's importance within QS&S algorithms but ruling it out as a research technique for algorithm construction and problem resolution in this work.

A different perspective for QS&S started when researchers working on quantum optimization algorithms were aware that executing a purely quantum algorithm to solve an optimization problem faster than a classical one was tremendously complex due to the slow evolution of hardware devices. Therefore, seeking a middle point between complex purely quantum algorithms with significant advantages but with questionable advantages, a new family of algorithms was conceived. These algorithms combining classical and quantum technology are known as quantum-inspired algorithms.

Within the quantum-inspired category, there are many types of algorithms: quantum algorithms executed on classical hardware, classical algorithms accelerated by small quantum hardware modules, algorithms that use quantum methods but are implemented entirely classically, or algorithms that combine quantum and classical modules, both hardware and software. This work specifically focuses on hybrid algorithms.

Hybrid algorithms, the central focus of this work, are currently generating high expectations to become the paradigm through which quantum computing enters the industrial sector. Under the umbrella of this concept, there are various algorithmic schemes, but the common thread is their modular nature, which combines both classical and quantum modules.

Hybrid algorithms have different approaches, but the main ones are twofold. On one hand, hybrid algorithms can be considered an end in themselves. They are constructed following this paradigm because there will be operations where quantum computing will never be faster than classical, and thus, it is always better to combine both. In this approach, quantum processing units (QPUs) are developed as processing modules that are added to classical supercomputers, similar to RAM or GPU processors. On the other hand, hybrid algorithms can be viewed as an intermediate step that must be taken because quantum devices are not yet mature enough to support the entire quantum algorithm. As technology develops, the classical modules of these

hybrid algorithms may be phased out.

This work aligns more with the philosophy of the second approach, which is the development of hybrid algorithms because quantum devices currently lack the capacity to analyze the size of the problem intended for resolution. However, the algorithms are designed with sufficient scalability so that once quantum hardware becomes available, it will be possible to execute the entire algorithm in a quantum manner.

Taking this intermediate step through classical-quantum hybrid computing is a way to try to gain quantum advantages in optimization problems much earlier. NISQ computers, specifically designed for such algorithms, are expected to be available sooner than other paradigms of quantum computers like early fault-tolerant computers. This implies the possibility of gaining an advantage in certain problems in the medium term. However, large-scale quantum computers capable of executing purely quantum algorithms are not expected until the long term, without specifying a date.

In this first chapter on quantum search & sample, a hybrid algorithm will be proposed that combines quantum walks technology to create a quantum Metropolis-Hastings algorithm capable of solving optimization problems strongly connected to industrial needs. In the following sections, various use cases where the explained hybrid algorithm has been applied will be discussed.

4

Quantum Metropolis Solver (QMS)

Following the philosophy of hybrid algorithms as an intermediate step for future fully quantum algorithms and looking for an algorithm that can be executed in the short term in NISQ, the quantum Metropolis Solver (QMS) algorithm is proposed. This algorithm follows the line of the Metropolis-Hastings quantum algorithm explained in chapter 3 and modifies it to create a software tool that can be executed following the universal computing paradigm.

As explained in the previous sections, optimization problems have always been considered to be of great appeal for quantum algorithms. These problems are complex for classical algorithms and, moreover, require almost exhaustive searches. QMS seeks to solve optimization problems by making use of the advantages of universal quantum computation of circuits. Therefore, the designed tool is fully executable by code in both quantum simulators running on classical hardware and purely quantum hardware following the circuit paradigm.

4.1 Motivation

QMS design is strongly influenced by the types of QS&S algorithms explained in chapter 3. It is an algorithm that combines search & sample techniques to arrive at a solution, and is fully applicable to both search & sample problems. To do this, QMS performs two operations iteratively, first sampling the circuit by performing a step, similar to W in the Equation 16, and then repeating these steps, W, a specified number of times until the probability of measuring the correct state is sufficiently increased.

Since the core of the QMS tool is the QM-H algorithm, the quantum advantage of this tool derives from this technique. Therefore, the main motivation to build QMS has been to find an algorithm based on QM-H that would allow a real implementation, keeping the same quan-

tum advantage and that could solve a wide range of QS&S problems, ultimately optimization problems.

Considering that one of the main motivations of this work and this thesis was to create realistic and implementable QC algorithms, QMS has been implemented and tested in the Python programming language and using the Qiskit quantum computing library. In addition, the QMS code and its modifications to the use cases have been published in public repositories on GitHub. Links can be found in each of the releases.

By open-sourcing the code, another motivation behind this work has been pursued, to enable any user to have an easy-to-use software tool that solves optimization problems through quantum computing algorithms, providing a straightforward and interpretable result. To run the code, it is only necessary to provide a simple description of the problem, and QMS will return the minimum/maximum energy configuration for their problem.

Finally, QMS has been designed with a completely modular philosophy, allowing any software component to be modified or adapted to a specific use case. Moreover, this promotes the hybrid model, as any component can be switched from quantum to classical and vice versa. In the following sections, three use cases for QMS are presented: quantum artificial intelligence, space exploration, and quantum chemistry.

The idea behind these three diverse use cases is to demonstrate that QMS can be applied to each of them and to an even broader range of optimization problems. The only common characteristic among them is the need for their fundamental operations to be quantum search & sample algorithms. Additionally, there is an intention to open a new line of research, following the hypothesis that delving deeper into these optimization algorithms is necessary through universal computing and not dedicated hardware such as quantum annealing.

4.2 Methodology

QMS operates by providing a description of the problem to be solved and specifying the precision in terms of the number of qubits with which the states should be represented. The problem description consists of defining the state space of the problem and the evaluation function to be applied to each state, guiding the search towards the state of maximum or minimum energy.

Ideally, this problem description would be an input file that defines two functions: one to generate states from an arbitrary state (inter-state transition function) and one to calculate energies for any state (evaluation function). However, considering the current state of quantum computing, QMS is currently designed so that both functions are implemented directly in the code. Depending on the use case, modifications are required to be made to the QMS code to

adapt it to the particular problem. This approach requires a thorough understanding of the problem to be solved. It is also important to realize that defining the problem using an input file with two functions does not require additional research; it would simply imply that the transition and evaluation functions would be executed in the quantum circuit, which implies a high processing and storage capacity for a quantum computer that is not currently available.

- $|\cdot\rangle_S$: State register, representing the entire state space. This register uses a binary encoding of a set of v variables. Each variable has a precision of q qubits and therefore can represent 2^q states. In total, the state space size represented by this state register is $2^{v \times q}$
- $|\cdot\rangle_M$: Movement register, it is composed by two independent registers, move id register $|\cdot\rangle_{move-id}$ that contains information about which variable register is going to be modified and move value register $|\cdot\rangle_{move-value}$ that contains if the modification of the variable is going to be plus or minus 1.
- $|\cdot\rangle_C$: Coin register, similar to that proposed by [71]. One qubit to represent the coin.
- $|\cdot\rangle_A$: Probability acceptance register, it is a register in which the value energy associated to each state is loaded. It is used to load the move acceptance probability on the coin.

These registers are used in a modified QM-H operator U_{QMS} to fit the new registers.

$$U_{QMS} = RV^\dagger B^\dagger FBV. \tag{17}$$

The operators that form U_{QMS} are similar to those explained in chapter 3, with differences in the implementations explained in publication 4.3.

An extra operator E, could be added. E is the energy operator that loads or calculate the energies associated to each state represented by $|\cdot\rangle_S$. This operator can be used in two ways. The theoretical approach is for QMS to be a fully quantum algorithm and calculate the energies associated with each state in superposition without the need to measure the states. This first case is the ideal, but as explained earlier, quantum hardware is far from making this a reality. The second way to use this operator follows the philosophy of hybrid algorithms. For this, a module is designed in classical computing capable of calculating the energies of quantum states. These energies are passed to the operator E, which reads and loads them into the probability acceptance register $|\cdot\rangle_A$ using a technique called QRAM [79, 80].

Another important aspect of QMS is its ability to be configured with different hyperparameters to adapt it to the specific problem being executed. For example, the number of times the operator U_{QMS} is applied, the precision to represent the probability acceptance register, or the

beta parameter (β). Specifically, the β parameter is one of the main configuration hyperparameters of QMS, as it regulates the balance between exploration and exploitation in the search. A low value of β favors exploration, while a high value favors exploitation. In practice, properly adjusting these hyperparameters is crucial for achieving good results in solving optimization problems.

β is defined as $\frac{1}{T}$ being T the effective (fictitious) temperature of the system. This concept is similar to the simulated annealing explained in subsection 3.4. The canonical use of T is high temperature at the beginning to focus on exploration and, progressively, reduce the temperature to find the absolute minima/maxima.

QMS allows for modification of the temperature T by adjusting β in each iteration. This enables the use of various annealing schedules, ranging from a constant schedule with a fixed β to an exponential reduction of β in each iteration, and including more gradual reductions with linear or geometric schemes.

Another key aspect of the quantum software tool, QMS, is the comparison with its classical counterpart, the M-H algorithm. For this purpose, a small module has been implemented within QMS that enables the execution of the classical algorithm on the same problem and under the same conditions. This allows benchmarking the results of the quantum algorithm and assessing whether it outperforms the classical one.

However, this comparison between classical and quantum is not straightforward. The main issue is deciding which metric to compare. Comparing execution times does not make sense because the quantum algorithm runs on a simulator on classical hardware. Counting the number of gates does not make sense either because quantum and classical gates are very different. Therefore, the metric chosen for comparison is Time To Solution (TTS).

TTS is a metric explained in [71]. It is a figure of merit that assigns a numerical value to the cost of executing a specific algorithm. In the case of QMS, it returns a numerical value that allows the comparison of both executions. TTS stands for time; therefore, the higher the TTS, the longer the time to solution and the worse the performance. Thus, in the comparison, the aim is to determine which algorithm has a lower TTS and, therefore, better performance.

TTS is calculated following the formula:

$$TTS(t) := t\frac{\log(1-\delta)}{\log(1-p(t))}, \tag{18}$$

where t is the number of time steps, equivalent to W. δ is the success probability, and $p(t)$ is the probability of hitting the ground state after t steps. Then the TTS is represented in a plot,

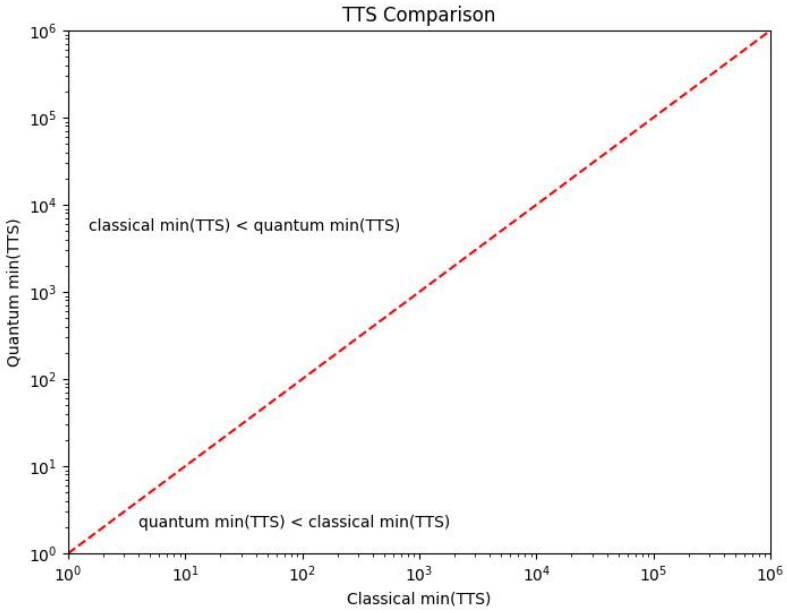

Figure 15: TTS comparison plot is divided by a red dashed line. This line splits the plot in two triangles: the upper triangle contains points with low classical TTS and high quantum TTS, which means classical advantage region. In contrast, lower triangle contains points with low quantum TTS and high classical TTS, which means quantum advantage region. Therefore, the desired behavior is that all points fall into the lower triangle.

shown in figure 15, with quantum TTS on the y-axis and classical TTS on the x-axis. This plot is divided in the diagonal by a dashed line, indicating the frontier between classical and quantum TTS. This dashed line creates two triangles, the upper triangle contains the points with low classical TTS and high quantum TTS, which means classical advantage region. In the opposite, the lower triangle contains values with low quantum TTS and high classical TTS, quantum advantage region. Ideally, all points should be in the quantum advantage region. Nevertheless, the quantum hardware is not ready yet for this situation. For that reason, some points will be in the classical region and others will be in the quantum advantage region. This makes necessary to represent the axis in logarithmic scale and to make a scaling exponent analysis. Using a square mean regression, it is possible to observe the tendency of the points. If the exponent is lower to 1 means that, with more TTS, bigger sizes of problems, all points will tend to be under the dashed line, so in the quantum advantage region. It is important to understand this concept in order to interpret correctly the result of the use cases.

With the development and testing done on QMS, a behavior has been observed that has been compared with other works. With small optimization problems, it is not possible to gain

quantum advantage because the execution of quantum circuits requires a certain computational overhead. The quantum advantage is expected to be observed with larger problem sizes.

In Figure 16, a plot with three different regions can be seen. The x-axis represents the size of the problems, and the y-axis represents the time needed to solve them. The three regions correspond to small, medium, and large problems. This plot is a simplification of the results that have been observed and compared with those of other authors. With small problems, classical algorithms have better performance, lower execution time due to the computational overhead of quantum circuits. This gap remains constant for a certain problem size. When that gap starts to shrink, it is considered a transition to medium-sized problems. In this new region of problems, the constant computational overhead begins to have less impact on performance, and the gap starts to close. In the last region, when equality is achieved between the performance of quantum and classical algorithms, it is clear that the trend for classical algorithms in complex optimization problems is to present a cliff in performance. The time required by classical algorithms increases exponentially. However, the expected performance of quantum algorithms grows more steadily, entering regions of practical quantum advantage. Of course, this figure is only an idealization of reality, but has many similarities with what has been observed in the use cases that will be explained later.

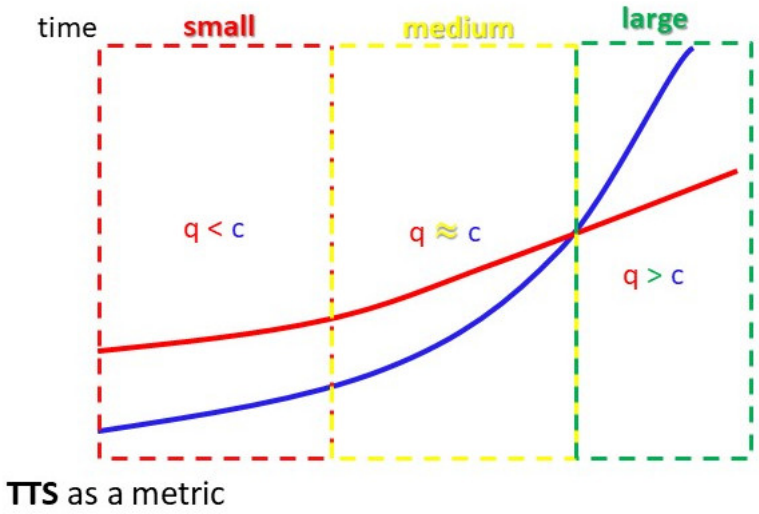

Figure 16: This figure represents three different sizes of problems solved by QMS: small, medium and large. The boundaries of the three regions are delimited by an inflection point in the ratio between **quantum** and **classical** algorithm performance. Between the small and medium size region, the inflection point is that the gap between quantum and classical starts to narrow because the computational overhead is no longer so important. Between the medium and large size region, the tipping point is that both algorithms have similar performance and the quantum starts to be better off.

Regarding the scalability in the third region of Figure 16, which pertains to large problems, two scenarios can be analyzed. First, if the number of states is exceedingly high, and secondly, if the differences in the evaluation function value between states are very low. In such cases, it becomes necessary to apply a large number of W operations.

In the first scenario, where a large number of states are involved in the problem, thereby demanding a significant number of qubits to represent them, the scalability of QMS is primarily determined by the number of qubits required for the state representation. Other registers, such as those for the coin, move value, and ancillas, either remain constant or increase logarithmically with the number of qubits needed for state representation. Therefore, the focus is primarily on examining how the number of qubits for the state register grows. In QMS, this growth follows a polynomial pattern concerning the number of variables required to represent the states, multiplied by the number of qubits necessary to represent all possible values of these variables.

If there are 100 variables and each variable is discretized using 20 qubits, QMS will require $100 \times 20 = 2000$ qubits to represent all possible states, resulting in 2^{2000} possible states. If a new variable is added, the system will require an additional $1 \times 20 = 20$ qubits to represent the states. Thus, the representation scales polynomially, as the number of variables multiplied by the number of discretization qubits. Although it is difficult to establish a comparison with the classical counterpart of QMS in representation scalability terms, the memory to represent 2^{2000} possible states will scale exponentially due to the necessity of representing them sequentially.

In the second scenario, where the differences in evaluation values between states are minimal, necessitating numerous repetitions of the W operator, it is crucial to examine how the number of W repetitions grows to find the solution. Due to QMS is a heuristic method which its performs depends on the problem, it is not possible to give an exact number, just estimations based on simulations. After the simulations of different use cases that will be explained in the next chapters, it is known that QMS requires to execute operator W up to an exponent of 4 number of the number of qubits, n_q. Therefore, while the number of possible states grows as 2^{n_q}, the upper limit of the number of W repetitions grows as n_q^4. The number of states grows much faster than the number W operators. It is important for the comparison of the number evaluations that a classical algorithm needs to search in the whole state space.

Comparing the number of evaluations that the classical and quantum requires to find a solution, the classical M-H algorithm requieres to explore between 60% and 80% of the total number of states [81]. Of course, it is a rough number that depends on multiple factors. However, it is easy to see that the classical m-h requires exploring a very large region of space to converge and this scales with the number of states. QMS, on the other hand, requires a number of repetitions of W that scales much more slowly and, by definition, at each repetition explores the entire state space in superposition.

The scalability conclusions shown here can be applied to all qms use cases because they are all based on the same algorithm derived from quantum walks.

To obtain the results of the use cases, QMS has been designed to run on both simulators and quantum hardware. All results from the use cases have been obtained using Qiskit's simulators. Additionally, in the use case of quantum chemistry, protein folding, a reduced version of QMS was executed for this specific use case on IBM's quantum hardware. This demonstrated that QMS is entirely valid for running in realistic environments.

With the ability to run on quantum hardware, coupled with QMS's good scalability in problem representation and algorithm execution, this tool could be considered as part of an industrial pipeline with quantum advantage. To reach that point, the only current limitation is the development of quantum hardware, which, as seen in previous sections, is undergoing rapid progress.

4.3 Results

- ✓ A quantum tool has been designed capable of solving real optimization problems and using real data. It is called QMS (Quantum Metropolis Solver)
- ✓ Implementation of software of a modified version of QM-H with the necessary adaptation of registers and operators.
- ✓ Implementation of hybrid architecture that can be tested in a early fault-tolerant quantum computer.
- ✓ Polynomial quantum advantage demonstrated in an important optimization problem.
- ✓ General software tool that can solve a wide range of optimization problems, just by receiving a description of the problem as input.
- ✓ Software tool with better scalability than a classical tool for optimization problems.
- ✓ Modular implementation that allows to modify any component of QMS to adapt it to a different problem. In addition, it was developed under an open source philosophy and published on GitHub.

Quantum Machine Intelligence (2023) 5:28
https://doi.org/10.1007/s42484-023-00119-y

RESEARCH ARTICLE

Quantum Metropolis Solver: a quantum walks approach to optimization problems

Roberto Campos[1,2] · **P. A. M. Casares**[1] · **M. A. Martin-Delgado**[1,3]

Received: 12 December 2022 / Accepted: 11 June 2023

Abstract

The efficient resolution of optimization problems is one of the key issues in today's industry. This task relies mainly on classical algorithms that present scalability problems and processing limitations. Quantum computing has emerged to challenge these types of problems. In this paper, we focus on the Metropolis-Hastings quantum algorithm, which is based on quantum walks. We use this algorithm to build a quantum software tool called Quantum Metropolis Solver (QMS). We validate QMS with the N-Queen problem to show a potential quantum advantage in an example that can be easily extrapolated to an Artificial Intelligence domain. We carry out different simulations to validate the performance of QMS and its configuration.

1 Introduction

Optimization problems are frequently encountered in various contexts, such as determining the optimal selection of products to buy in the supermarket, based on their quantity and quality within a given budget (known as the knapsack problem Bretthauer and Shetty (2002)), identifying the best combination of public transportation options to minimize commute time (a variant of the Traveling Salesman Problem, TSP Hoffman et al. (2013)), or selecting the most interesting landmarks to visit a city during a limited vacation period (goal oversubscription problem Smith (2004)). These types of complex optimization problems are commonly faced by different industries in their routine operations. However, unlike daily optimization problems, industrial optimization problems often require significant computational resources to achieve an optimal solution because of the large number of possible combinations that need to be evaluated.

Optimization problems of special interest to the industry often involve multiple variables with a large number of dimensions and complex optimization functions, making it challenging to evaluate each possible configuration of problems. Typically, the function that is optimized is a valuable resource that can have significant economic implications for both individuals and companies. Examples of these types of problem include the routing problem, which involves identifying the optimal path between multiple locations Bektaş and Laporte (2011); Kumar (2012); Toth and Vigo (2014), portfolio optimization in finance, which requires balancing risk and reward when deciding which products to buy and sell Markowitz (1968); Rubinstein (2002), and the protein folding problem, which aims to minimize energy by rotating the protein structure Kryshtafovych et al. (2019).

A common phenomenon for which these problems suffer is the "curse of dimensionality" Bellman (1956); Kuo and Sloan (2005) defined by Bellman. It occurs when the dimensionality of the data grows rapidly, leading to a significant increase in the volume of data. As a result, the data become scattered and difficult to cluster, which poses a significant challenge for problem solving.

The scattering of data caused by this phenomenon makes brute-force solutions impractical for larger optimization problems, as the size of the problem grows exponentially Kolaitis and Thakur (1994). Despite the fact that the procedure for solving larger problems only checks a minimal subset of all possible combinations, it is still necessary

Roberto Campos
robecamp@ucm.es

P. A. M. Casares
pabloamo@ucm.es

M. A. Martin-Delgado
mardel@ucm.es

[1] Departamento de Física Teórica, Universidad Complutense de Madrid, Madrid, Spain

[2] Quasar Science Resources, Las Rozas de Madrid, SL, Spain

[3] CCS-Center for Computational Simulation, Universidad Politécnica de Madrid, Madrid, Spain

Published online: 17 July 2023

to evaluate numerous potential combinations. The selection of an appropriate technique to solve optimization problems can significantly affect the time required to obtain a solution. For example, it may be possible to reduce the time required to solve a problem from centuries to hours or days. This transformation can make a seemingly intractable problem solvable.

Optimization problems have been extensively studied from a theoretical perspective, and numerous toy problems have been developed as simplified versions of real-world problems with fundamental similarities. These problems serve as benchmarks for testing the performance of new algorithms on easily executable instances. Examples of such benchmark problems include the TSP Hoffman et al. (2013), the knapsack problem Bretthauer and Shetty (2002), and the N-Queen problem Gent et al. (2017), which are similar to the routing problem, the risk/reward finance problem, and the problem of selecting the best action to execute, respectively.

The fundamental aspect of optimization problems is the requirement to traverse numerous states with an associated value until the state with the minimum value is reached. The trial and error process, followed by refinement, can be automated using computer simulations to accelerate the problem solving process. Consequently, it is crucial to establish an accurate correspondence between the real-world problem and the simulation.

Due to the complexity explained above, most complex optimization problems belong to the NP complexity class Crescenzi et al. (1995). Thus, polynomial advantages are often the best one may hope to attain. Quantum computing is a natural approach, based on the fact that quantum walks can achieve a quadratic speed-up in hitting time over their classical counterparts Montanaro (2015).

2 Optimization algorithms and random walks

There is an extra difficulty in dealing with optimization problems, the representation. Some of them, such as the connections between cities in the traveling salesman problem (TSP), can be easily represented on a computer. However, others, such as modeling the air around the wing of an airplane, pose a more significant challenge, necessitating the use of simplified models. In certain cases, oversimplification may even be required due to the intractability of the original problem. This was historically the case with lattice models of protein folding Robert et al. (2021).

There are some different representation options to convert the problem into an algorithm-solvable instance. In this work, a four-element representation is chosen. The elements are as follows:

- States: Possible values the system can take for a given problem.
- Transitions: Possible states generated from each state.
- Evaluation function: Function to calculate the reward/value of each state.
- Goal: Objective of the problem, minimize or maximize the evaluation function.

In order to introduce the quantum algorithm that we have selected in this work, we have to explain some classical algorithms first. Classical random walks are not only very powerful, but they also form the basis for very widely used Monte Carlo algorithms, routinely used for optimization problems. The Monte Carlo method consists of a random sampling of state space to approximate a function. It works better with a larger population because the error classically decreases as $1/\sqrt{N}$Daniell et al. (1984). The most relevant aspect of the Monte Carlo method is that it serves as a basis for optimization algorithms.

A related technique also based on random walks and inspired by statistical physics is the simulated annealing algorithm Kirkpatrick et al. (1983). The core concept is a search algorithm that always accepts transitions that lower the energy, but with a certain probability, it also accepts transitions to higher energies. The probability of moving to a higher energy state at the beginning of the execution is high because the simulated temperature starts warmer. However, as steps are executed, the algorithm goes cooler and the probability is reduced. That process helps the algorithm explore many states at the beginning, avoiding local minima. Because of this, the algorithm converges slowly to the minimum energy state.

Combining random sampling of the Monte Carlo method from a probability distribution and guided stochastic search of random walks and simulated annealing results in an algorithm called Metropolis-Hasting Metropolis et al. (1953); Hastings (1970). It is used to approximate a probability distribution π_x by mixing it with a random walk W until an equilibrium is reached, $W\pi_x = \pi_x$.

The Metropolis-Hastings algorithm requires three methods: (i) a procedure to sample initialization states, (ii) a procedure to propose state transitions, and (iii) an evaluation function that scores how good is a given state. The latter is often called "energy" E due to its connection to statistical physics. It will determine the acceptance probability of the transition proposed at point (ii), $\min(1, \exp(-\beta\Delta E))$, where β is a parameter called inverse temperature.

There exists a quantum version of random walks. Quantum walks can also be understood as a generalization of Grover's algorithm Grover (1997). The first proposal, by Ambainis Ambainis (2004), was restricted to Johnson graphs. Soon, Szegedy presented bipartite quantum walks generalizable to any ergodic chain problem Szegedy (2004).

Both can be shown to offer Grover-like quadratic speedup in the hitting time, in other words, in the time required to find the marked item. The latter quantum walk has been widely used, for example, in the context of Quantum Metropolis algorithms Temme et al. (2011); Montanaro (2015). Also, Szegedy's proposal has found a variety of applications Paparo and Martin-Delgado (2012); Paparo et al. (2013, 2014); Kadian et al. (2021).

Most of the previous quantum Metropolis algorithms assumed a slowly changing parameter β and often phase estimation to evolve the state from the uniform superposition to the target stationary distribution Somma et al. (2008); Yung and Aspuru-Guzik (2012). However, this is different from the way classical algorithms operate, where β is changed much more rapidly, and no additional techniques other than random walks are used. For this reason, Lemieux et al. proposed a quantum version of the Metropolis-Hastings algorithm that makes use of quantum walks heuristically, similar to how random walks are used classically Lemieux et al. (2020).

In this work, we discuss and analyze the behavior of the quantum Metropolis-Hastings (M-H) algorithm in an optimization problem that arises in the field of Artificial Intelligence (AI). To test the M-H algorithm and facilitate other users to use quantum M-H, we implemented a software tool called Quantum Metropolis Solver (QMS). Our tool uses as input a description of an optimization problem and generates the minimum-cost solution. Also, it has additional functionalities such as plotting, classical solution comparison, and deep analysis of quantum M-H algorithm solutions. The tool is open-source philosophy-oriented and was coded in Python using Qiskit modules. The tool code is public in https://github.com/roberCO/QMS-OSS.

QMS application is threefold. First, it can be used as a metric tool to test the performance of the quantum M-H algorithm in a concrete search problem. Additionally, QMS can be integrated into hybrid classical-quantum algorithms since QMS input is a classical problem description, but it can generate classical or quantum output. Finally, we compare classical and quantum variants to computationally assess potential quantum advantages.

3 The Metropolis-Hastings algorithm: classical vs. quantum

The M-H algorithm is a Markov chain procedure because the transition probabilities depend only on the current state, and it is a Monte Carlo technique due to the generation of a random sequence of samples from a probability distribution $g(x)$. The result of these two properties is a Markov chain Monte Carlo algorithm, able to rapidly mix and generate low-energy states.

The Metropolis-Hastings algorithm is used to sample the stationary state of the Markov chain π_x. As a starting point, a uniform random sample x is generated. Then, new samples (x') are generated from the previous sample using some generation function $g(x'|x)$ and accepted according to their energy differences. If the energy of x' is lower than x, it is accepted; otherwise, an acceptance probability is calculated using an evaluation function $f(x)$ as $\alpha = f(x)/f(x')$. This process is similar to a random walk with steps dictated by W, preparing a stationary state π_x. Its stochasticity allows the algorithm to explore a larger search area and avoid getting caught in local energy minima. Figure 1 shows a scheme of this algorithm.

The M-H algorithm is an optimization technique with great utility in domains ruled by a minimization or maximization function and a well-known probability distribution to generate successors. Particularly interesting are problems with uncertainty in the output, in which the solution of the problem is not known, and the task is to find the lowest value/energy state. For example, finding the hyperparameter configuration of a deep learning architecture. In contrast to problems like routing, in which the final point is known and the task is to find the path between the initial and final point given by the problem definition.

The distinguishing advantage of M-H over other heuristic algorithms is the rapid generation of successors due to its stochasticity and its unique dependence on the previous state Zabinsky (2009). On the other hand, that high-speed generation may penalize the M-H algorithm with a poorly guided search far from the optimal, thus the M-H algorithm is better suited for highly complex and unstructured configuration spaces where the stochastic state generation is advantageous.

The process of finding a solution by iteratively trying different steps to refine the current state to an acceptable solution is an old and well-known process. It has been one of the foundations of human reasoning and problem solving. However, the big change comes when there are machines capable of performing thousands of these refinement steps quickly or even at the same time. Using this fast step execution, the M-H algorithm takes the brute-force philosophy and incorporates Markov theory to achieve an algorithm capable of testing many states with a minimum guided search. For this reason, one of the strengths of M-H lies in the ability to evaluate successive states quickly.

3.1 The classical MH method

The performance of the classical M-H algorithm is strongly influenced by two factors: mixing time (MT) of the Markov chain defined in Eq. 1 and Monte Carlo error. These two aspects determine the number of necessary steps to obtain an acceptable solution, and they are key in creating a quantum

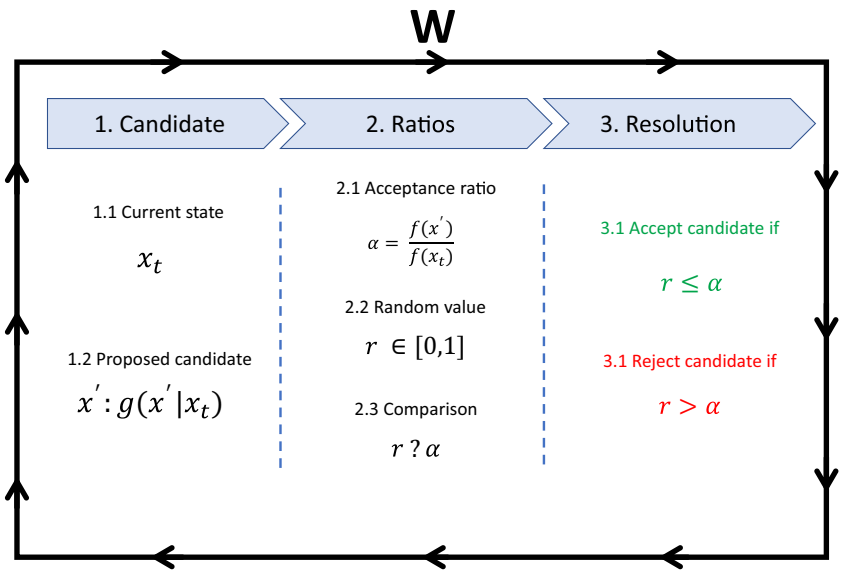

Fig. 1 This figure shows all steps of the Metropolis-Hastings algorithm. The search for the minimum energy state starts in the state x_t. Then, the first step is to propose a new candidate x' related to x_t by the distribution function $g(x_t)$. Once the candidate is generated, it is necessary to calculate a numeric value α to see how much better/worse it is over the current state. Currently, a random number r is generated between 0 and 1. Then, both values α and r are compared. If α is greater than or equal to r, the change is accepted and x' becomes in x_t; otherwise, x' is discarded, and the new x' will be calculated again in the same x_t. This process is repeated until a fixed number of w is reached or the evolved distribution, over time, s_0 is less than a value of ϵ, different from the objective distribution π

version that enhances both. The mixing time of the Markov chain is the time it takes for a Markov chain to reach a stationary distribution. Ref. Boyd et al. (2004) explains the importance of this value optimization. In contrast, the Monte Carlo error is the distance between the calculated distribution at step N and the stationary distribution. In Ref. Wolff and Collaboration (2004), there are examples to reduce the error of the Monte Carlo methods.

The Mixing Time $MT(\epsilon)$ can be interpreted as the minimum number of steps t of the Markov chain that should be applied to any initial distribution (π_0), such that the result is ϵ-close to the stationary distribution (π), under distance D (Eq. 2).

$$MT(\epsilon) = \min\{t | \forall t' > t, \forall \pi_0, D(P^{t'}\pi_0, \pi) < \epsilon\}, \tag{1}$$

Distance $D(p, q)$ between probability distributions p and q is in turn defined as the sum of probability differences at each vertex of the Markov chain, under those distributions,

$$D(p, q) = \frac{1}{2}\sum_{v=1}^{N} |p_v - q_v|. \tag{2}$$

In this work, we have focused on an M-H algorithm that converges to the lowest energy state. However, since the successor generation in M-H is governed by a probability distribution function, it can also be used to sample the unknown stationary distribution of a Markov chain. The algorithm can generate samples from the probability distribution, which can be used to approximate a probability density function Yildirim (2012). Specifically, it can be applied to problems that prohibit complete enumeration of all paths Flötteröd and Bierlaire (2013). Again, this M-H sampling can be quantum versioned naturally, since the quantum circuit of the M-H can be executed repeatedly until getting a distribution of the result of each execution shot.

As explained above, the limiting factor to getting a speed-up with the M-H algorithm is the mixing time and the error reduction. Going one step further, it is possible to identify three points that we can optimize in the M-H execution: reduce the number of evaluated states, avoid getting stuck at local minima, and evaluate states faster. Quantum computing can help at these points as the eigenvalue gap of the quantum walk is quadratically smaller than the classical eigenvalue gap as explained in Magniez et al. (2011) with the formula $\Delta = \Omega(\delta^{1/2})$ being δ the eigenvalue gap of the classical walk and Δ the phase gap of the quantum walk. Although there are classical approaches to solve this problem Calderhead (2014), Grover's algorithm and quantum walks add amplitudes instead of probabilities, and their difference ends up showing as a quadratic speed-up Szegedy (2004).

3.2 A quantum version of the MH method

A quantum version of the Metropolis-Hastings algorithm exploits a reduced complexity due to a smaller eigenvalue gap compared to its classical counterparts. This fact helps to infer the minimum energy state quicker. The essence of this advantage comes from the application of the quantum walk operator to an initial uniform superposition of all possible states so that the number of steps to mix the chain is reduced (Table 1).

Quantum walks can be understood as a generalization of Grover's algorithm Galindo and Martin-Delgado (2000).

With two Grover-like reflections, Szegedy Szegedy (2004) constructs a quantum walk on a bipartite graph. However, in this work, we substitute the bipartite graph with a coin $|c\rangle$ via an isomorphism Lemieux et al. (2020), which creates an entanglement with the states $|s\rangle$. This produces a quantum walk $|\Psi\rangle = |s, c\rangle$, where the states are represented as a superposition $|s\rangle$ of possible states.

$$|s\rangle = \left\{ \sum_{x \in \mathbb{N}} \alpha_x |x\rangle \right\} \in \mathcal{H}_s, \tag{3}$$

and the coin ($|c\rangle$) is in the coin space $\mathbb{C}^2$

$$|c\rangle = \{\alpha_1|\uparrow\rangle + \alpha_2|\downarrow\rangle\} \in \mathcal{H}_c. \tag{4}$$

In Lemieux et al. (2020), a Szegedy quantum walk is used as a basis to construct the circuit of a quantum Metropolis-Hastings algorithm. In this work, we show the implementation complexity of W in a quantum circuit using unitary operators. The main challenge is the application of the operator and its inverse (unitary operator) in each W because the inverse of the operator depends on whether the change is accepted or not. In the ideal case, it would be necessary to apply a conditional inverse operator in each step. In the M-H algorithm, after a change is proposed, there are two options: accept or reject a proposed change, and this duality is a key problem in creating a unitary operator. The solution proposed by Lemieux et al. Lemieux et al. (2020) is a different unitary operator for W that is isomorphic to the original Szegedy walk operator U_W and is represented as

$$\tilde{U} = RV^{\dagger}B^{\dagger}FBV, \tag{5}$$

where R is the reflection operator, V is the move preparation operator, B is the coin operator, and F is the spin-flip operator. The operators F, B, and V can be seen as a direct quantization of the classical operations in the M-H algorithm of proposing a candidate and accepting/rejecting it depending on whether it minimizes or maximizes the value function. On the other hand, R is the only pure quantum operator for the quantum M-H algorithm that is used to generate Grover-like rotations with the potential to exhibit a polynomial advantage. All of these operators simplify the implementation to a circuit, if we compare with Szegedy's implementation of W.

Table 1 Algorithm of Quantum Metropolis-Hastings, detailing how the new candidate is proposed and the acceptance probability and random number are calculated. This algorithm executes several steps W

Algorithm 1: Metropolis-Hastings algorithm

Initialize $x_t \sim g(x)$
for *iteration step = 1,2,...,* W **do**
  Propose: $x' \sim g(x'|x_t)$
  Acceptance Probability
  $\alpha(x'|x_t) = min\{1, \frac{f(x')}{f(x_t)}\}$
  Random variable r: $r \in [0, 1]$
  if $r \leq \alpha$ **then**
    Accept the proposal: $x_{t+1} \leftarrow x'$
  else
    Accept the proposal: $x_{t+1} \leftarrow x_t$
  end
end

4 QMS: Quantum Metropolis Solver software tool

We present a software framework whose core is the circuit proposed by Lemieux et al. (2020) and build a library around the Metropolis-Hastings quantum algorithm, allowing any user to solve optimization problems with the quantum M-H algorithm. We call this software architecture Quantum Metropolis Solver (QMS). We implement it in a quantum simulator running on a classical computer. QMS pretends to be one step beyond the evolution of quantum walks and quantum M-H algorithm.

There are numerous papers on quantum walks. In our case, we want to continue a development path that we believe has two main previous points (and many intermediate ones), the proposal of quantum walks by Szegedy and the quantization of the M-H quantum algorithm by Lemieux et al. Starting from the proposed $\tilde{U}$ operators, we have modified this algorithm so that it can solve optimization problems. For this, we have implemented all the circuits in the qiskit library Aleksandrowicz et al. (2019) and modified it to be able to read data from an oracle. We have also modified the ways of initializing the circuit, optimized certain operations and we have included the option of working with different betas that allow testing the algorithm with a quantum walk out-of-equilibrium. We have created an M-H quantum model capable of solving any optimization problem defined as a

set of states and values associated to these states. Our main contributions are as follows:

- **Software tool:** We give the community easy-to-use software to solve problems in which Metropolis-Hastings has a proven advantage.
- **Study of Quantum Metropolis-Hastings to an optimization problem related to Artificial Intelligence:** We provide evidence that the quantum advantage of quantum walks and the QM-H algorithm can be applied to Artificial Intelligence to optimize the search process inherent in any AI technique. The case study to validate this idea is the N-Queen problem, which has been used recurrently and is still used, as a benchmark for new classical AI algorithms.
- **Scaling law:** We analyze the performance of QMS by finding the scaling law for the N-Queen problem. In this way, we add another example of the application of the quantum MH algorithm to the previously analyzed case study of the Protein Folding problem Casares et al. (2022).
- **Comparison of different implementations of quantum walks:** We compared three different implementations of quantum walks and quantum Metropolis-Hastings on the N-Queen problem. The proposal of Szegedy (2004) uses a bipartite graph to search in the state space. This algorithm was later modified in Ref. Lemieux et al. (2020) with discrete operators and substituting the n-dimensional search with a binary search performed with a coin. Finally, we also compared the results with an operator ordering different from the qubitization method in Low and Chuang (2019).
- **Quantum walk out-of-equilibrium:** In QMS, we included the option to have different beta β schedules. This β appears in Eq. 6 and is related to the probability of acceptance of the proposed change. The point of varying the beta value during the quantum walk execution is to have a quantum walk out-of-equilibrium and get the Markov chain mixed faster than a conventional quantum walk.
- **Oracle:** We designed a subroutine to load all energy/value data into the quantum walk. It allows us to execute larger problems with precalculated values.
- **Deltas:** We preprocess all values associated with the states to reduce the complexity of the circuit and the qubits needed to represent them. We calculate the difference between the state values and encode it into ancilla registers of the oracle.
- **Initializations:** We designed our software tool to create quantum circuits for different initializations of the problem. For example, we can start with an equiprobable distribution in the states, even if the number of states is not a power of 2, or with a function defined by the user. We can work with problems with or without boundary conditions, etc.

The underlying motivation for implementing QMS is to give the scientific and industrial community a test-bed to solve optimization problems with quantum algorithms, including a comparison with its classical counterpart. This software tool abstracts the inherent complexity of implementing quantum algorithms. Users can define a problem to solve with just an input file. In this file, the problem is defined as a set of states and associated costs. Then, QMS will return the state with the least cost as a solution. It is assumed that the states are connected among them in a graph so that a Markov chain algorithm can be applied.

QMS has been designed not only to offer a final result, but also to show statistics that help to understand the performance of the algorithm. There are three possible outputs of QMS: the minimum cost state, the TTS result, or the probability distribution. Stressing the point of a test-bed, QMS allows a TTS comparison between the quantum algorithm and its classical version. In such a mode, the same problem is executed in a classical Metropolis-Hastings algorithm and a quantum one, and both minimum TTS achieved by each is shown. Furthermore, a plot with the TTS curve is generated as a function of t in (9) for both the quantum and the classical M-H algorithms.

The classical M-H module for comparison with quantum M-H has also been implemented by us. It is exactly the counterpart of the quantum Metropolis-Hastings. We implemented a classical random walk following the M-H algorithm to decide if the change is accepted or not. It takes the same input as the quantum walk and performs a search with the same number of steps (Ws). Then, the classical M-H is executed n times to also get the probabilities of having one state or another as a result. Therefore, we execute the classical M-H w by n times.

Our software tool, detailed in Fig. 2, has been thought of as a user-friendly library that simplifies its use. The user simply defines the problem in an input JSON, a file that stores simple data structures and objects in a standard format. Then, it is possible to configure the QMS execution with a configuration file where some parameters are defined. For example, the number of steps, which is equivalent to the number of times that operator W is applied, can be tuned just by modifying the value of the parameters *initial_step* and *final_step*. Any number of steps in between will then be analyzed.

The core of QMS is an implementation of the circuit proposed in Lemieux et al. (2020) with some optimizations and modifications. From Eq. 5, the operators are the same:

- V is *move_preparation*: This operator creates a superposition of all possible state transitions.

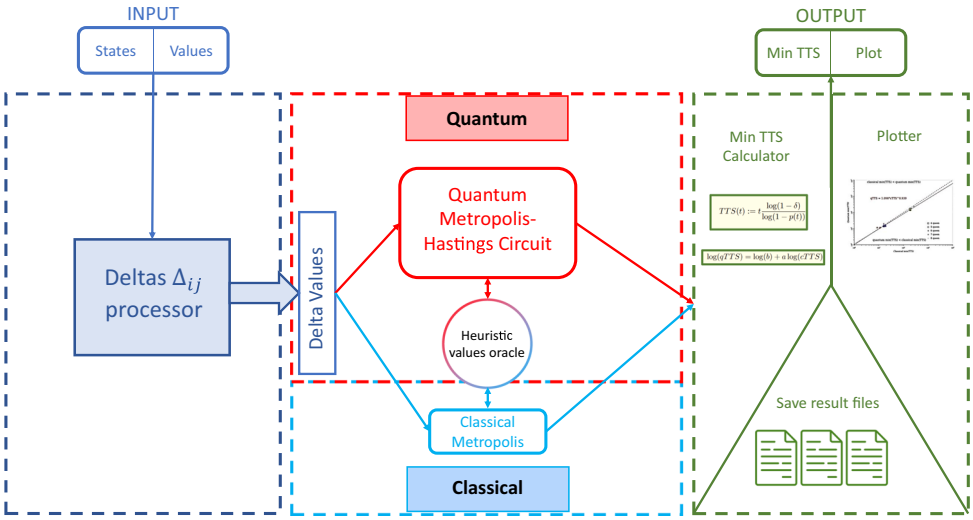

Fig. 2 Scheme of the QMS architecture. It receives a JSON file with a description of the possible states and the values associated with them. Then, the deltas between states (Δ_{ij}) are calculated and sent to the quantum M-H module. This QM-H module constructs an initialization circuit to obtain the initial state x_t. In addition, the operator circuit W is generated many times with a fixed parameter. Once the circuit is created, QM-H executes the circuit. The results obtained after execution using raw amplitudes are processed to get a plot with the evolution of the TTS, a numeric TTS value, or the probabilities. In parallel with the quantum M-H execution, QMS has the option to execute a classical M-H, to compare both versions of the M-H algorithm. This classical M-H module has the same structure and connections with input and output as its quantum counterpart

- B is *coin_preparation*: This operator creates the superposition of the coin and rotates the coin in those states where the change is accepted.
- F is *conditional_move*: This operator performs the change in the states in which the coin has been rotated. This is the M-H acceptance/rejection step.
- R is *reflection*: Similarly, a Grover reflection is used for amplitude amplification of the marked states.

However, the representation of the states (also called register) done in previous work has been optimized. Previous work used a representation of N qubits to represent M possible registers to move, being $N = M$, so one qubit for each register. With this representation, all registers had a 0 value except the register to move that had a 1. We substitute this unary representation in the move register with a binary representation, using only $log_2(M)$ qubits, where M is the number of registers. That change allows us to reduce the number of qubits in the whole system. Furthermore, the heuristic that guides the algorithm is based on an inverse temperature parameter β. The acceptance value comes from the condition:

$$A_{ij} = \min\left(1, e^{-\beta(C_j - C_i)}\right), \tag{6}$$

where A_{ij} is the acceptance ratio of the proposed changed, $\beta = 1/T$, C_j is the candidate cost, and C_i is the old state cost. It implies that if the cost of the candidate is lower, the change is accepted. Otherwise, the change is only accepted with exponentially decreasing probability in the inverse temperature.

The input file of QMS is a set of tuples (state, cost). States are represented with a binary notation starting from 0 and cost is a real number. An example tuple could be [(101): 65.53], 101 is the states in binary notation, and 65.53 is the cost.

For internal representation, QMS requires $\lceil \log_2(n) \rceil$ qubits to represent n input states. The cost associated with each state is not directly represented; on the contrary, what is stored in QMS is the cost difference between the connected states, represented as Δ. Δ_{ij} is set to 1 if the cost of candidate state j is lower than the cost of the state i. Otherwise, Δ_{ij} stores a codification of the cost difference between the states, as shown in this equation:

$$\Delta_{ij} = \begin{cases} 1 & E_j < E_i, \\ e^{-\beta(E_j - E_i)} & E_j \geq E_i, \end{cases} \tag{7}$$

this codification corresponds to the probability of change considering the β of the step. Then, all probabilities are stored in a QRAM as an oracle that can be accessed in each W.

We define four registers to store all information in QMS:

- **States** $|\cdot\rangle_S$: Contains each possible state affected by operator move preparation (V).
- **Move** $|\cdot\rangle_M$: Contains the candidate move state affected by operator move preparation (V).

- **Coin** $|\cdot\rangle_C$: Coin affected by operator coin preparation (B).
- **Oracle** $|\cdot\rangle_O$: Contains the acceptance probability between all connected states affected by the conditional move of the operators (F) and the reflection (R).

The move register $|\cdot\rangle_M$ can be divided into two registers: move id register $|\cdot\rangle_{Mi}$ to know which state is going to be changed and move value register $|\cdot\rangle_{Mv}$ that indicates how the register is to be changed (+1, -1, swap left, swap right, etc.).

In any optimization problem, it is critical to choose the initial state distribution. The gap between the initial state and the goal state determines the execution time or the quality of the solution. QMS receives an input set of states from which it has to find the least energetic one. QMS takes an initial point and applies the W operator the number of times defined by the user. The state reached after the application of the W operators is the solution returned by QMS. For that reason, our library allows different initial states to leave to the user the freedom of selecting at which point the search should start. It is even possible not to select just a point but rather a probability distribution that will determine the initial point for the search. The preparation of such a probability distribution is carried out by a process called initialization.

- **Fixed:** QMS starts the search in a state selected by the user. This option is useful for refinement problems in which there is a first approximate solution, and the optimization consists of a search for a more quality solution close to the initial approximation.
- **Random:** QMS starts in a uniform superposition of states that is equivalent to a random state because no one has more probability than others to be selected.

QMS initialization can also be classified by how states are generated, as follows:

- **Sequential:** This mode generates candidates sequentially, simply adding or subtracting one unity from the coordinate. For example, if the problem is a pawn moving onto a chess board, the states could be represented as the row and column occupied by the pawn (column, row). So, if the pawn is in (4, 5), one possible candidate is to add 1 position in the column, resulting in a state (5, 5).
  - **Circular:** This option allows connecting frontier states with periodic boundary conditions. For example, if the previous pawn is in position (5, 7), so it is in the upper row, it would be possible to add one row to place the pawn in the lower row, state (5, 0).
  - **Non-circular:** This option does not allow connecting frontier states, so it is configured with non-periodic boundary conditions. For example, if the pawn is in the row frontier (5, 7), it is not allowed to sum 1 in rows.
- **Swap:** This mode generates candidates exchanging coordinates, and it does not allow collisions. For example, if there are 3 queens placed on the board and each queen is in one column, such that there are no two queens in the same column. The state will represent the row of queens. One possible state is (1, 0, 2), first queen in row 1, second queen in row 0, etc. The candidates are generated by exchanging queen positions; for example, a candidate switching the first and second queens would result in (0, 1, 2), which means the first queen in row 0, the second queen in row 1, etc.

QMS is a software tool with a white-box design and a modular architecture. It means that it is made up of an aggregation of modules that receive input, process it, and serve the result to other modules. These modules can be analyzed, modified, or even replaced by other modules with the same data interface following similar input/output format rules. This open and flexible architecture is an advantage that can enhance the use of the tool by the community because it is easy to understand, including new functionalities that may be added in the future or fixing bugs. QMS architecture is detailed in Fig. 2.

The whole architecture has been implemented using Python3 because it is an open-source language and very used by the developer community. The quantum module calls Qiskit and Qiskit-Aer for the quantum simulations on the classical computer. Qiskit is also an open source Python module that simplifies circuit creation and simulation, and it has proven good performance with large circuits with more than 25 qubits and multi-thread execution, as can be seen in Fig. 11 of Suzuki et al. (2021).

5 Case study: Quantum Artificial Intelligence

Any Machine Learning (ML) algorithm modifies its internal state and the world representation following a deliberative process. This process is based on reasoning about the input data to obtain a knowledge model of the problem. During the reasoning process, the system generates different hypotheses to explain the environment and execute the correct action. For example, adjusting connection weights in an artificial neural network or modifying the value of the action in a Reinforcement Learning agent.

The hypothesis generation and selection steps require a fast search in the state space (hypothesis space) to evaluate

which of all hypotheses available fits better with the problem to be solved. This search process is one of the bottlenecks of any ML algorithm, and quantum computing can speed it up. For that reason, we consider QMS to help in the Artificial Intelligence domain, improving the performance of the search process. We decided to validate our tool on a problem that has been extensively used as an AI benchmark, the N-Queen problem. Since this problem is in essence a search problem with multiple similarities with the search of an AI algorithm, any algorithm that gets a good performance in N-Queen can be easily adapted to other AI problems based on search, as a search of hypothesis in an ML algorithm.

The N-Queen problem is a spin-off of the classic chess problem Bowtell and Keevash (2021). The N-Queen goal state is a chess board of n rows and n columns with n queens such that no queen attacks the others. Therefore, there are no two queens in the same row, column, or diagonal as shown in Fig. 3. Many solutions have been proposed to the N-Queen problem Luria and Simkin (2021); Simkin (2021).

The N-Queen problem has been studied by many authors as a NP-complete problem Khan et al. (2009); Crawford (2016); Güldal et al. (2016). However, this categorization has not yet been clearly demonstrated Gent et al. (2017). The study that has the most detailed description of the complexity of the N-Queen problem places it in *beyond the P class* Hsiang et al. (2004). However, Gent et al. (2017) showed that N-Queen completion, a variation of N-Queen in which some queens are preplaced and the challenge is to place the others, is an NP-complete problem. This work was used by Torggler et al. (2019) to propose a quantum solution for a different variant of N-Queen, excluded diagonals, which is also NP-complete.

This kind of reasoning, searching for a hypothesis that explains and generalizes input data, can be seen as an abstraction of general knowledge from concrete examples, and it is known as inductive reasoning. As it is explained in Ref. Russell and Norvig (2010), the inductive reasoning goal is to find the hypothesis with the best balance between the classification of the existing examples and the generalization of the new examples. The search of this hypothesis with the best reward generalization/classification is equivalent to the N-Queen search, and, for this reason, N-Queen is a typical problem used as a benchmark in classic AI papers Gent et al. (2017). For example, this problem was used to introduce the backtracking search Walker (1960).

During environment interaction, any rational agent generates a set of hypotheses or associations from input data, which is similar to human learning from the environment or describing a problem. Then, it searches for which hypothesis is the best to explain the world. This process is known as a search in a decision tree. Sometimes, the agent selects one hypothesis and adapts it to new input data, but this is similar to a search of variations around the fixed point performed by QMS. Both symbolic learning (e.g., using first-order logic) and non-symbolic (e.g., using weights in a neural network) require the hypothesis space search to find the best next decision to take.

That process is similar to most machine learning algorithms. As Mitchell describes it in Ref. Mitchell (1997), the process of learning can be understood as a searching task in a large space of hypotheses and the search for the hypothesis that best fits the training and upcoming examples. This is the main reason why if it is possible to speed up the search process using quantum computing, it is possible to get advantages in AI algorithms. The reason is that what is commonly called the "learning process" is just the process of searching for the best explanation (hypothesis) for the data received, trying to be as ready as possible for future data entering the system.

Since the final state of the problem solved by QMS is unknown because it has uncertainty in the output (the task is to find an unknown configuration with minimum cost), the QMS algorithm makes a state-space search guided by the objective of minimizing a cost function. As Knuth describes, there is a type of searching based on comparing keys (values) of states Knuth (1998). In our case, we perform an informed

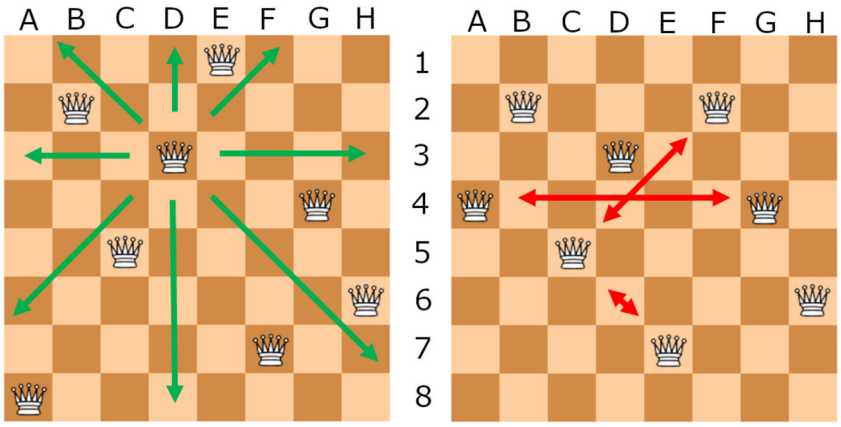

Fig. 3 N-Queen problem explanation with 8 queens in a chessboard of 8×8. On the left is a valid solution for the problem because no queens are attacking one another as is explained for queen in D3. On the right is an example of the same chess board but with 4 queens attacking each other (A4, C5, E7, F6, and G4)

search (guided by a heuristic) that compares the different heuristic values of the states and a sequential search because it is necessary to perform the search moving the queens in a certain order. Of course, the sequential concept is just a formalism to describe the search because quantumly it is done in superposition. These are the reasons why we view QMS applied to N-Queen as a search algorithm.

An AI agent does a similar search to optimize its interaction with the environment to speed up its goal attainments. Due to the similarities between the search in hypothesis space and the QMS search around an initial state (initial hypothesis), it is possible to understand QMS as a technique that implements one of the most fundamental steps in any Artificial Intelligence agent.

Another feature of the N-Queen problem solved with QMS is the transition probabilities. Since we are solving the problem with a technique based on Metropolis-Hastings, the probability of transitioning to one state or another from the current state is of particular relevance. According to Eq. 6 and Eq. 7, the probability will always be one that the heuristic value is reduced (the number of attacks between queens is lower) and a value between 0 and 1, otherwise. This probability between 0 and 1 allows QMS to be able to transit to a new state even if the heuristic value increases, avoiding local minima. The probability value is calculated proportional to the difference in the heuristic value between the current state and the next state. If the difference of the heuristic value is small, the probability will be close to one; otherwise, it will be close to zero. In addition, this probability can be regulated using a coefficient of β.

As explained in Section 4, QMS requires an input file which is a set of tuples (state, cost). In the N-Queen problem, the states are all possible combinations of boards that can appear for a given n. To reduce the size of the state space, we represent each state as a list with n positions (one per queen), and each position represents the row of the queen. Assuming that it is not possible to have two queens in the same column, a movement is just a permutation in the position (row) of two queens. The scheme shown in Fig. 4 will be represented with the list (0, 1, 2, 3) indicating that the first queen is in row 0, the second one in row 1, etc.

This representation is an efficient codification of the problem that reduces the number of states. Possible movements are swaps between positions. For example, in Fig. 4, the first and second queens are swapped between them. Once the codification of the board has been explained, it is necessary to explain the heuristic value associated with each position on the board. The heuristic that we designed counts the number of attacking queens and penalizes them with extra cost if one queen attacks more than one other queen. Heuristic (H) is defined by Eq. 8, and the objective is to minimize the heuristic value. If the heuristic is 0, this board configuration is a solution.

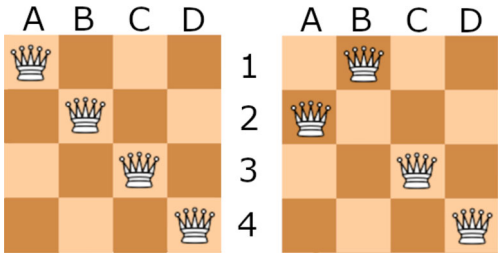

Fig. 4 This state for $n = 4$ is represented, in the left as (0,1,2,3) because the first queen is in row O (A1), second queen is in row 1 (B2), etc. In the right, a swap between the first and second queen was executed, and the resulting state is (1, 0, 2, 3) because the first queen is in row 1 (A2), the second queen is in row 0 (B1), etc

$$H = \sum_{i=0}^{n} \sum_{j=i+1}^{n} [\delta_{row_i,row_j} + \delta_{diag_i,diag_j}] * \gamma, \tag{8}$$

where the first sum counts the heuristic value for all queens, and the second sum counts the heuristic value for each queen. δ_{row_i,row_j} is 1 if $queen_i$ and $queen_j$ are in the same row (or same for diagonal), else it is 0. γ is an accumulative value that counts the number of queens that $queen_i$ is already attacking. It is a multiplicative factor, which is increased by 1 for each extra attacked queen. $\gamma = 1$ for the first attacked queen by $queen_i$, $\gamma = 2$ for the second attacked queen by $queen_i$, etc. This heuristic returns a high penalty for boards in which a queen is very badly placed, such that, it is attacking multiple other queens, which it is something it is necessary to avoid.

In the N-Queen problem, the number of solutions for each size n determines its complexity. A higher number of solutions implies more goal states and a more guided search because the gradient between states is bigger and the transition to the goal state is faster. Results in faster solution generation. In Table 2 extracted from Number solutions of the n-queen problem (2022), it is possible to see that $n = 6$ has significantly fewer solutions than the previous case, $n = 5$, and the next case, $n = 7$. This affects the complexity of the problem. As can be seen from the simulations plot in Fig. 5, the TTS obtained for $n = 6$ and $n = 7$ is similar, despite the fact that for $n = 7$, the state space is much larger than for $n = 6$.

6 Simulation results

To validate the QMS tool with the N-Queen problem, we execute different simulations with both classical and quantum Metropolis-Hastings algorithms. These simulations were

Table 2 Table of number of solutions for N-Queen problem by n. In general, the number of solutions is increased with the number of n, except in $n = 6$ that it is reduced which affects the complexity

n	Number solutions
4	2
5	10
6	4
7	40

executed using the framework *Qiskit* Aleksandrowicz et al. (2019) and the included free noise simulator *QASM*. The N-Queen problem requires many qubits to represent its states, and the actual simulator has reduced capacities to represent large amounts of data. For that reason, we calculated the number of qubits that our codification needs and the memory consumption in the *QASM* simulator for the number of qubits. In Table 3, the number of qubits necessary for each register in the QMS software tool is detailed.

The list of necessary registers is as follows:

- **Coordinates** register represents each state. It is necessary to codify each state in binary form, so the number of qubits is the number of registers multiplied by the qubits necessary for the binary representation.
- **Move id** register represents the coordinate to move, so it is an index that requires a binary representation of the number of registers.

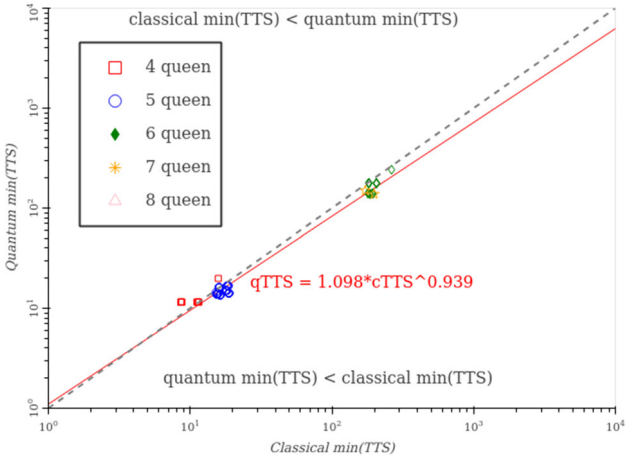

Fig. 5 Comparison between the classical and quantum TTS for N-Queen problem with n=4, 5, 6, and 7 with 74 samples of the problem. The dashed gray line separates the spaces of the classical advantage (upper triangle) and the quantum advantage (lower triangle). The key aspect to notice in this figure is that for n = 4, most points are in the classical advantage region, but when the problem increases its difficulty, most points are in the quantum advantage region. The scaling exponent is 0.939

Table 3 The number of qubits for each register in the QMS software tool

N-Queen codification	
Coordinates	$n * \lceil \log_2(n) \rceil$
Move id	$\lceil \log_2(n) \rceil$
Move value	1
Coin	1
Ancilla	3
Total	$n * \lceil \log_2(n) \rceil + \lceil \log_2(n) \rceil + 5$

- **Move value** register indicates whether the movement is up or down (left or right in the swap case). It only requires one qubit.
- **Coin** register represents the binary decision of acceptance or rejection of the proposed candidate.
- **Ancilla** register is used to store the change probability of the proposed candidate. It can be represented with 3 or more qubits depending on the selected precision in probability.

Table 4 represents the number of qubits for each size n in the N-Queen problem. This table is useful to understand the complexity of the circuit and the necessary resources to execute it on a classical computer. In our case, we have 128 GB of RAM available, but we only have results of $n = 7$ due to the execution times that are around 2 weeks per instance of the problem with $n = 7$.

We execute the simulations using the TTS metric, explained in Eq. 9, as a figure of merit. We test QMS for n =4, 5, 6, and 7. The decision to stop at $n = 7$ is directly related to the time consumption of each execution. To get more cases of the problem with different initial configurations, we slightly modify the N-Queen rules. In each instance, we fixed one queen in one position, considering that this queen is stuck in the position for any reason. It is common to have this kind of restriction in an ML problem as, for example, a mandatory point to visit in a route generated with Deep Learning. This new rule gives us extra points to evaluate our problem.

Table 4 The number of qubits and memory RAM consumption of *QASM* simulator for each size n instance problem

	QMS	
N	Qubits	RAM Memory
4	15	0.1 GB
5	23	0.8 GB
6	26	1.5 GB
7	29	8 GB
8	32	65 GB

Without this N-Queen modification, we would only have 4 different samples of the problem, one per n value. In the simulation, we have 74 different instances of the N-Queen problem with 9 instances for $n = 4$, 42 instances for $n = 5$, 21 instances for $n = 6$, and 2 instances for $n = 7$. The reduced number of instances for $n = 7$ results in 4 weeks of execution.

The metric proposed and used by Lemieux et al. (2020) is called Time To Solutions and denoted as TTS. It is a figure of merit that measures the expected number of steps required to find a solution. It is helpful to compare procedures that need to be repeated in case of failure, like this sampling algorithm. TTS strikes a balance between probability increase and the number of steps in each execution, which means that lower TTS implies less expected execution time.

$$TTS(t) := t\frac{\log(1-\delta)}{\log(1-p(t))}, \tag{9}$$

where t is the number of steps executed, δ is the success probability, and $p(t)$ is the probability of hitting the ground state after t steps. With this metric and a scaling law exponent analysis, Lemieux et al. got a polynomial speedup of 0.75, e.g., classical TTS $= O(\text{quantum TTS}^{0.75})$, arguing that their proposal scales better than the classical Metropolis-Hastings and can thus be advantageous in bigger problem instances. The exponent indicates how the relationship between the classical and quantum algorithms scales. Lower than 1 means that quantum complexity scales more favorably than classical. We can estimate this exponent with a linear least-square fitting in the logarithmic scale for both classical and quantum minimum TTS. Since we want to see the scaling law exponent of quantum TTS against classical TTS, we follow the equation $y = bx^a$, x and y, classical ($cTTS$) and quantum ($qTTS$) TTS, respectively. In logarithmic scale:

$$\log(qTTS) = \log(b) + a\log(cTTS), \tag{10}$$

being a the exponent to define the scaling between $qTTS$ and $cTTS$. This exponent defines three regions:

$$a = \begin{cases} > 1 & \text{quantum TTS} > \text{classical TTS}, \\ 1, & \text{quantum TTS} = \text{classical TTS}, \\ < 1, & \text{quantum TTS} < \text{classical TTS}. \end{cases} \tag{11}$$

The results are shown in Fig. 5. This plot shows the relationship between classical and quantum TTS. It is divided into two triangles by a gray dashed line. The upper triangle shows the classical advantage region where the majority of $n = 4$ points are and the lower triangle with quantum advantage where all the points of $n = 6$ and $n = 7$ are. The plot shows that the resulting points present a tendency to move toward the region of quantum advantage as the problem size n increases, and we can conclude that there is a possible quantum advantage of quantum M-H against classical M-H. We can quantify it using a linear least-squares fitting with an exponent a (defined in Eq. 11) of 0.939.

We also test the core of QMS, quantum walks, to find the best performing algorithm. Due to the discretization carried out by Lemieux et al. in the quantum M-H unitary operator $\tilde{U}$, we test whether the sorting of the operators could affect the results. We also include two other alternative sorting options in the comparative. We define Preparation with the operators VB, Selection with the operator F, Inverse Preparation with $B^\dagger V^\dagger$, and Reflection with R.

- **Lemieux et al.**: Preparation-Selection-Inverse Preparation-Reflection. Namely, it corresponds to the sorting:

$$\tilde{U} = RV^\dagger B^\dagger FBV. \tag{12}$$

- **Qubitization**: Explained in Low and Chuang (2019). Inverse Preparation-Reflection-Preparation-Selection. Namely, it corresponds to the sorting:

$$\tilde{U} = FVBRB^\dagger V^\dagger. \tag{13}$$

- **Alternative**: Selection-Inverse Preparation-Reflection-Preparation. Namely, it corresponds to the sorting:

$$\tilde{U} = VBRB^\dagger V^\dagger F. \tag{14}$$

To compare these three different sorting options, we execute the N-Queen problem with $n = 4$, 5, and 6 including several instances of fixed queens. For each sorting and each n, we get the mean and standard deviation. Using these parameters, it is easy to compare whether the TTS have significant differences between them. We show the results in Fig. 6. The operator sorting election selected by Lemieux et al. achieves a lower TTS value for all the problem sizes tested. In addition, it is possible to observe a similar tendency between the different sorting options, but separated by a gap. Thus, these simulations show that the Lemieux et al. sorting works better.

6.1 Time complexity analysis

In order to prove the superiority of the quantum Metropolis-Hastings algorithm, we need to carry out a detailed analysis of several factors. It is possible to see two clearly differentiated point clouds in Fig. 5, one corresponding to the instances of 4 and 5 queen that are small size instances of the problem and the other corresponding to 6, 7, and 8 queen corresponding to medium size instances. We observe that quantum algorithms have a small computational overhead over classical ones, which penalizes them with small optimization problem instances. For example, in Fig. 5, points corresponding

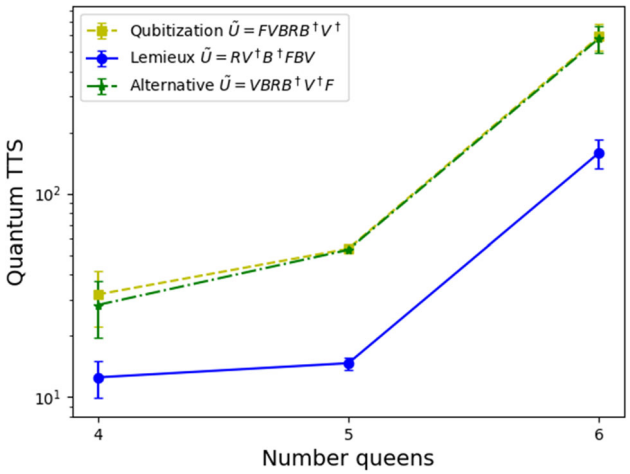

Fig. 6 This figure shows the three different sorting options that we test for the quantum walk operator W in eqs. 12, 13, and 14. In blue, the sorting proposed by Lemieux et al. 12 gets a minimum TTS lower than the other two sorting options for $n = 4$, $n = 5$, and $n = 6$. It is possible to observe a similarity between the evolution of W in eq. 12 and the other two sorting options in 13 and 14

to the 4-queen problem are in the classical advantage region (above the dashed line) and points in the 5-queen problem are in a transition region between the classical and quantum advantage region (over the dashed line). This penalization due to computational overhead is compensated with the bad classical scalability for larger problems (6, 7, and 8 queen instances) for which all points fall in the quantum advantage region.

Since we are conducting simulations with small and medium size problems, we run the risk that the quantum advantage will be hidden by the quantum computational overhead of small problems. To avoid that, we perform a scaling exponent study. This exponent α is a linear least squares fitting on the logarithmic scale of both values (quantum and classical minimum TTS). In Eq. 11, the fitting exponent analysis shows how it is expected that QMS behaves with larger instances of the problem (< 1).

Theoretically, an algorithm based on quantum walks has a quadratic advantage as an upper bound. Before the simulations, we calculated a range of the exponent value between 0.5 and 1. After the simulations, we observed an exponent of value 0.939 which being less than 1 indicates that there is a quantum advantage that will become larger as the size of the problem increases.

It is important to understand that we get this exponent based on our figure of merit, TTS. Therefore, we want to show that we are able to observe a quantum advantage by analyzing TTS. The first remark is that, if the exponent trend is sustained for larger problems, there are polynomial advantages; in fact, we observed that the exponent is improved (lowered) each time we test a larger problem.

The maximum problem size allowed by the quantum simulator is 2^{12} (8-Queen). However, using classical computers, N-Queen has been solved for much larger instances. For example, if we want to solve the 60-Queen problem, the state space using our representation would be 2^{180}. If we want to analyze the difference using TTS between classical and quantum algorithms, we need to know how TTS grows in both cases according to the problem size ($\log(problem_size)$ *vs* $\log(TTS)$). For classical TTS, the scaling factor is 7.89034, and for quantum, it is 7.13895, so the relation between them is 0.904. Extrapolating these values to the 60-Queen problem, we get a difference of TTS equivalent to 40.551 faster the quantum over the classical M-H solution.

7 Conclusions and outlook

We have studied quantum optimization algorithms to test how they could challenge existing classical algorithms for industrial problems. Classical optimization algorithms have been contributing to find solutions to hard problems that are otherwise impossible to solve by computers with brute force. However, classical optimization algorithms have limitations in scalability and can be optimized with quantum computing. Specially, we have focused on the Metropolis-Hastings algorithm that has many applications for industrial problems and its quantum counterpart, the quantum Metropolis-Hastings.

A quantum version of M-H was proposed by Szegedy (2004) and modified to obtain an implementable version by Lemieux et al. (2020). We use both works to construct a software tool that has the M-H algorithm at the core and which can solve any optimization problem given in simple format as a list of state-cost tuples. This tool is an easy-to-use Python module that receives a description of the problem and returns the minimum cost position. Besides, QMS evaluates to return a Time To Solution (TTS) value of the search to find the solution. This TTS metric is a figure of merit in evaluating the performance of the tool and comparing it with the classical algorithm.

As we have shown, it is possible to use QMS with any combinatorial optimization problem that has uncertainty in the output. Thus, the goal is to find an unknown state with some properties. This requirement is met in most optimization problems at the industrial level (knapsack problem, TSP, routing, etc.). The search process to find the state with the minimum cost in these problems converts them into an NP-complexity problem. It is in this family of problems in which quantum computing polynomial advantages can be the most useful, that is the reason why we selected them. The works by Szegedy and Lemieux et al. show a polynomial advantage using quantum walks, which we extrapolate to a general-purpose tool for any optimization problem.

We validate our QMS quantum software tool in the Artificial Intelligence domain. It is well known that one of the bottlenecks in Machine Learning algorithms is the search process that the algorithm performs to find the explanation that best fits the input data. This search is very similar to the process used to solve an optimization problem, as has been explained in the literature Russell and Norvig (2010); Mitchell (1997). Therefore, we consider that QMS could be applied to speed up some processes of an Artificial Intelligence algorithm.

Since we want to show how QMS can help AI algorithms using quantum search, we identify the N-Queen problem as a good benchmark for this task. The N-Queen problem is considered a benchmark for AI Gent et al. (2017); Russell and Norvig (2010) and can also be solved by a quantum algorithm, as we show in this work. In the simulations, we observe that the quantum algorithm gets better results than its classical counterparts. We also perform an analysis of the scaling with a linear least-squares fitting, getting an exponent of 0.939. Although the classical Metropolis-Hastings algorithm is not the state-of-the-art procedure to solve the N-Queen problem, we emphasize that our goal is to show the quantum advantage that QMS can get in a search problem. This N-Queen problem case study for QMS is added to the case study we proposed in a previous paper Casares et al. (2022), also with quantum advantage.

Another study we carried out was to understand why the discrete operators were sorted in a non-standard way Lemieux et al. (2020), and we have also compared the TTS results for different sorting options. We again used the N-Queen problem as a benchmark to test the $\tilde{U}$ operator defined by Szegedy Szegedy (2004), Lemieux et al. Lemieux et al. (2020), and Low et al. Low and Chuang (2019).

Finally, future work should include more case studies to test QMS tools and possible applications. It would be interesting to analyze in multiple domains if the exponent is always above 0.5, which is the value required for a quadratic advantage.

Acknowledgements R.C. and P.A.M.C contributed equally to this work. We acknowledge support from the CAM/FEDER Project No.S2018/TCS-4342 (QUITEMAD-CM), Spanish MINECO grants MINECO/FEDER Projects, PGC2018-099169-BI00 FIS2018, PID2021-122547NB-I00 FIS2021, the "MADQuantumCM" project funded by Comunidad de Madrid and by the Recovery, Transformation, and Resilience Plan - Funded by the European Union - NextGenerationEU and Ministry of Economic Affairs Quantum ENIA project. M. A. M.-D. has been partially supported by the U.S. Army Research Office through Grant No. W911NF-14-1-0103. P. A. M. C. thanks to the support of a MECD grant FPU17/03620, and R.C. the support of a CAM grant IND2019/TIC17146.

Author Contributions R.C. and P.A.M.C contributed equally to this work. M. A. contributed and supervised the work.

Funding Described in the acknowledgments.

Data availibility The code has been placed on the GitHub platform (https://github.com/roberCO/QMS-OSS).

Declarations

Ethics approval and consent to participate Not applicable.

Human and animal ethics Not applicable.

Conflict of interest The authors declare no competing interests.

References

Bretthauer KM, Shetty B (2002) The nonlinear knapsack problem–algorithms and applications, European Journal of Operational Research, vol. 138, no. 3, pp. 459–472

Hoffman KL, Padberg M, Rinaldi G, et al (2013) Traveling salesman problem, Encyclopedia of operations research and management science, vol. 1, pp. 1573–1578

Bektaş T, Laporte G (2011) The pollution-routing problem. Transportation Research Part B: Methodological 45(8):1232–1250

Bellman R (1956) Dynamic programming and lagrange multipliers. Proceedings of the National Academy of Sciences 42(10):767–769

Kumar SN, Panneerselvam R (2012) A survey on the vehicle routing problem and its variants, Intelligent Information Management, 2012

Boyd S, Diaconis P, Xiao L (2004) Fastest mixing Markov chain on a graph. SIAM review 46(4):667–689

Bretthauer KM, Shetty B (2002) The nonlinear knapsack problem-algorithms and applications. European Journal of Operational Research 138(3):459–472

Calderhead B (2014) A general construction for parallelizing Metropolis-Hastings algorithms. Proceedings of the National Academy of Sciences 111(49):17408–17413
Casares PAM, Campos R, Martin-Delgado MA (2022) Qfold: quantum walks and deep learning to solve protein folding. Quantum Science and Technology 7(2):025013
R. Bellman, Dynamic programming and Lagrange multipliers, Proceedings of the National Academy of Sciences, vol. 42, no. 10, pp. 767–769, 1956
F. Y. Kuo and I. H. Sloan, Lifting the curse of dimensionality, Notices of the AMS, vol. 52, no. 11, pp. 1320–1328, 2005
Daniell G, Hey AJ, Mandula J (1984) Error analysis for correlated Monte Carlo data. Physical Review D 30(10):2230
Flötteröd G, Bierlaire M (2013) Metropolis-Hastings sampling of paths. Transportation Research Part B: Methodological 48:53–66
Galindo A, Martin-Delgado MA (2000) Family of Grover's quantum-searching algorithms. Physical Review A 62(6):062303
Gent IP, Jefferson C, Nightingale P (2017) Complexity of N-queens completion. Journal of Artificial Intelligence Research 59:815–848
Grover LK (1997) Quantum mechanics helps in searching for a needle in a haystack. Physical Review Letters 79(2):325
Daniell G, Hey AJ, Mandula J (1984) Error analysis for correlated Monte Carlo data, Physical Review D, vol. 30, no. 10, p. 2230
Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing, Science, vol. 220, no. 4598, pp. 671–680
Hoffman KL, Padberg M, Rinaldi G et al (2013) Traveling salesman problem. Encyclopedia of operations research and management science 1:1573–1578
Hsiang J, Hsu DF, Shieh Y-P (2004) On the hardness of counting problems of complete mappings. Discrete mathematics 277(1–3):87–100
Kadian K, Garhwal S, Kumar A (2021) Quantum walk and its application domains: a systematic review. Computer Science Review 41:100419
Ambainis A (2004) Quantum walk algorithm for element distinctness, in Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, FOCS '04, (USA), p. 22-31, IEEE Computer Society
Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
Temme K, Osborne TJ, Vollbrecht KG, Poulin D, Verstraete F (2011) Quantum metropolis sampling, Nature, vol. 471, no. 7336, pp. 87–90
Kolaitis PG, Thakur MN (1994) Logical definability of NP optimization problems. Information and Computation 115(2):321–353
Kryshtafovych A, Schwede T, Topf M, Fidelis K, Moult J (2019) Critical assessment of methods of protein structure prediction (CASP)-round xiii, Proteins: Structure. Function, and Bioinformatics 87(12):1011–1020
Kumar SN, Panneerselvam R (2012) A survey on the vehicle routing problem and its variants, Intelligent Information Management, 2012
Kuo FY, Sloan IH (2005) Lifting the curse of dimensionality. Notices of the AMS 52(11):1320–1328
Lemieux J, Heim B, Poulin D, Svore K, Troyer M (2020) Efficient quantum walk circuits for Metropolis-Hastings algorithm. Quantum 4:287
Low GH, Chuang IL (2019) Hamiltonian simulation by qubitization. Quantum 3:163
Lemieux J, Heim B, Poulin D, Svore K, Troyer M (2020) Efficient quantum walk circuits for Metropolis-Hastings algorithm, Quantum, vol. 4, p. 287
Magniez F, Nayak A, Roland J, Santha M (2011) Search via quantum walk. SIAM journal on computing 40(1):142–164
Markowitz HM (1968) Portfolio selection. Yale University Press
Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. The journal of chemical physics 21(6):1087–1092
Mitchell TM (1997) Machine learning. McGraw-Hill, New York
Montanaro A (2015) Quantum speedup of Monte Carlo methods. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 471(2181):20150301
Magniez F, Nayak A, Roland J, Santha M (2011) Search via quantum walk, SIAM journal on computing, vol. 40, no. 1, pp. 142–164
Paparo GD, Martin-Delgado M (2012) Google in a quantum network. Scientific reports 2(1):1–12
Paparo GD, Müller M, Comellas F, Martin-Delgado MA (2013) Quantum google in a complex network. Scientific reports 3(1):1–16
Paparo GD, Dunjko V, Makmal A, Martin-Delgado MA, Briegel HJ (2014) Quantum speedup for active learning agents. Physical Review X 4(3):031002
Casares PAM, Campos R, Martin-Delgado MA (2022) Qfold: quantum walks and deep learning to solve protein folding, Quantum Science and Technology, vol. 7, no. 2, p. 025013
Rubinstein M (2002) Markowitz's portfolio selection: a fifty-year retrospective. The Journal of finance 57(3):1041–1045
Suzuki Y, Kawase Y, Masumura Y, Hiraga Y, Nakadai M, Chen J, Nakanishi KM, Mitarai K, Imai R, Tamiya S, et al (2021) Qulacs: a fast and versatile quantum circuit simulator for research purpose, Quantum, vol. 5, p. 559
Bowtell C, Keevash P (2021) The N-queens problem, arXiv preprint arXiv:2109.08083
Luria Z, Simkin M (2021) A lower bound for the N-queens problem, arXiv preprint arXiv:2105.11431, 2021
Somma RD, Boixo S, Barnum H, Knill E (2008) Quantum simulations of classical annealing processes. Physical Review Letters 101(13):130504
Suzuki Y, Kawase Y, Masumura Y, Hiraga Y, Nakadai M, Chen J, Nakanishi KM, Mitarai K, Imai R, Tamiya S et al (2021) Qulacs: a fast and versatile quantum circuit simulator for research purpose. Quantum 5:559
Crawford KD (2016) Solving the N-queens problem using genetic algorithms, in Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's, pp. 1039–1047, 1992
Temme K, Osborne TJ, Vollbrecht KG, Poulin D, Verstraete F (2011) Quantum metropolis sampling. Nature 471(7336):87–90
Torggler V, Aumann P, Ritsch H, Lechner W (2019) A quantum N-queens solver. Quantum 3:149
Torggler V, Aumann P, Ritsch H, Lechner W (2019) A quantum N-queens solver, Quantum, vol. 3, p. 149
Russell S, Norvig P (2010) Artificial Intelligence: a modern approach. Prentice Hall, 3 ed
Wolff U, Collaboration A et al (2004) Monte Carlo errors with less errors. Computer Physics Communications 156(2):143–153
Mitchell TM (1997) Machine learning. New York: McGraw-Hill
Yung M-H, Aspuru-Guzik A (2012) A quantum-quantum metropolis algorithm. Proceedings of the National Academy of Sciences 109(3):754–759
Zabinsky ZB et al (2009) Random search algorithms. University of Washington, USA, Department of Industrial and Systems Engineering

5

S&S applied to quantum artificial intelligence (QAI)

Quantum computing promises great advantages over classical algorithms in difficult problems. One of the fields that has classically been characterized by having the most complex problems has been artificial intelligence (AI). In this category included all the problems in which algorithms had to process so much data that they had to be able to make decisions autonomously. Therefore, it is natural that quantum computing has also looked to this field to hybridize with it and create the concept of quantum artificial intelligence (QAI).

The concept of artificial intelligence, whether classical or quantum, encompasses many techniques. In this work, an AI algorithm is defined as one capable of running autonomously under changing conditions, doing some kind of reasoning or searching. Within the family of AI algorithms, three main groups can be distinguished: search algorithms, machine learning algorithms (ML) and multi-agent algorithms. In previous sections, search algorithms have been discussed in more detail, although they were not specifically dedicated to AI problems. In this section, ML algorithms will be discussed in more detail. An AI and ML general scheme is shown in figure 17.

ML algorithms take a set of input data and by reasoning learn from it to generate a knowledge model of the problem to find the required solutions. Within ML algorithms there are three other types of algorithms: supervised learning, unsupervised learning and reinforcement learning.

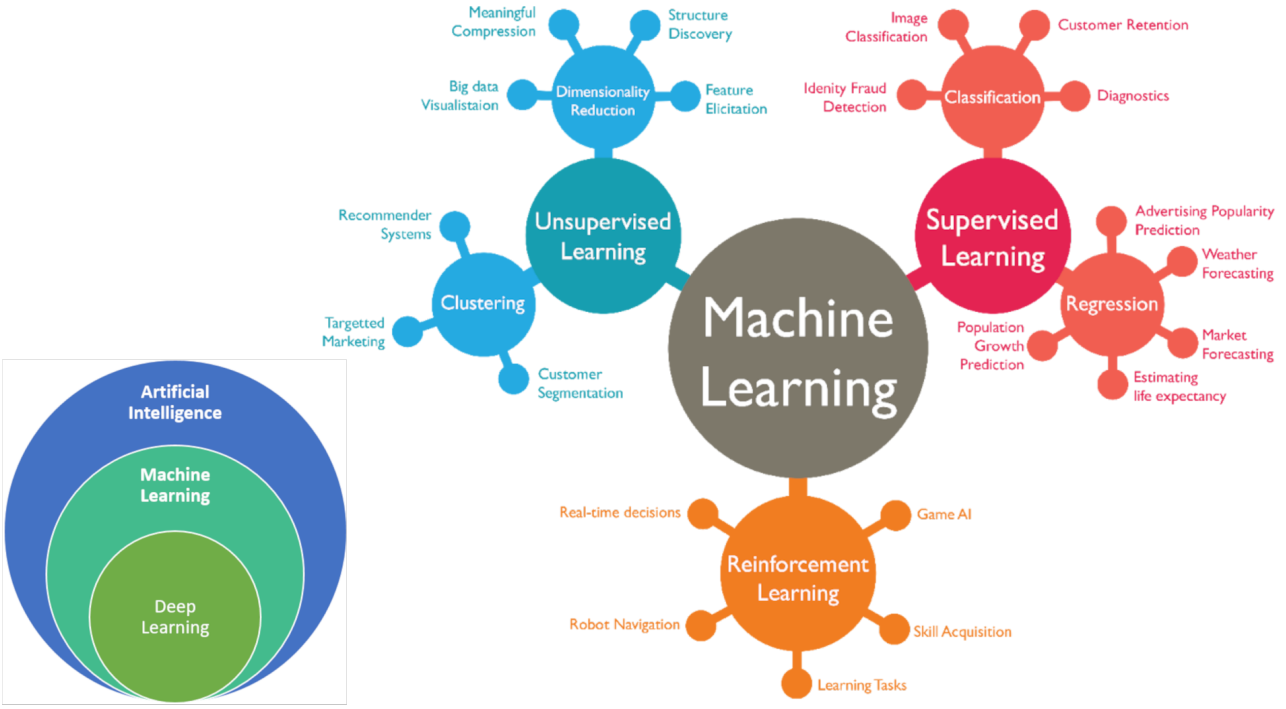

Figure 17: This figure shows the relationship between artificial intelligence, machine learning and deep learning in the left. In the right, there is a scheme of the different algorithms inside the machine learning field (extracted from [82]).

The main motivation for developing the field of QAI in general and QML in particular is to improve classical algorithms. This improvement can occur in different areas, but especially focuses on three, reduction of training or execution times, improvement of the accuracy of the algorithms, and reduction of energy consumption.

Currently, AI is in a moment of high growth and implementation in industrial technologies. However, its development limitations have already begun to become apparent. At the software level, they are beginning to see limitations in data processing that cannot be overcome [83]. Since these limitations cannot be overcome with new algorithms, the solution focuses on creating specific hardware that handles more data at higher speeds, for example, with dedicated AI-only chips [84]. Although this approach also has limitations, the size of the chips cannot be reduced below a certain threshold without quantum effects appearing. In fact, this limit has already been reached and no more transistors can be introduced on a single chip [85]. The solution is no longer to introduce more transistors on a chip to double the capacity every two years, as proposed by Moore's law [86]. Now, the solution consists of stacking chips and making computers with more

interconnected chips, giving the whole a higher capacity [87]. However, the concept of creating chips linked together in large supercomputers following the High Performance Computing (HPC) philosophy also has limitations and disadvantages. HPC computing centers require increasingly complex architectures for interconnecting the processing chips and, above all, they have very high energy consumption [88].

The first step in developing QAI algorithms, especially QML algorithms, has been to look for a source of inspiration. Just as happened with the first scientists who wanted to develop classical AI algorithms but did not know how to do it and started to mimic human brain processes, the development of QML has taken as a starting point classical algorithms that performed well and tried to make quantum versions [89].

This strategy is valid because the objective is clear, to have more efficient QML algorithms than the classical ones, but it is unknown how, so it was necessary to start the research at some point. However, unlike what happened in classical ML, which mimicking the human brain and learning proved to be very useful, such as neural networks, evolutionary algorithms or Reinforcement Learning, in the field of QML it seems to have led the research to a dead end.

Mimicking classical algorithms to create their quantum versions has proven to have limitations either because there are processes that cannot be reproduced in the quantum paradigm or because classical algorithms are so well-developed that it is impossible to beat them. Moreover, it does not seem a good solution to try to imitate classical algorithms because then the quantum advantage would fall only on the advantage that the quantum HW can offer, something that, currently, is not guaranteed that running a quantum circuit is faster than doing it classically.

At this point, pessimism seems to prevail regarding the field of QAI and QML in particular. Conceptual and fundamental problems have been encountered that have made it clear that the current trend followed so far in the field is not correct and must be redirected [90]. However, the opportunity offered by quantum computing to accelerate algorithms in general remains a fact, and it is clear that classical AI has limitations, so there remains an opportunity for quantum algorithms. Once the target is clear and relevant to research, it is necessary to open up new research lines. In this work, QS&S algorithms have been developed that can serve as the basis for a new line of QML research that diverges from the limited state of the art.

To explain how QS&S can be applied to the QAI field, the close relationship between search and artificial intelligence will be explained. Then it will be explained some QAI algorithms and the limitations that have been observed, and finally how QMS has been applied to QAI and what are its future applications will be explained.

5.1 Quantum S&S for QAI

There is a close relationship between the field of optimization and AI. Many of the algorithms used for optimization problems are the basis for complex AI algorithms. In fact, many of the basic AI processes can be seen as problems in which a certain parameter needs to be optimized. This situation can be seen in evolutionary algorithms trying to optimize a fitness function, multi-agent algorithms trying to optimize group behavior, or ML algorithms where a hypothesis has to be optimized to achieve a balance between generality and expressiveness or a parameter optimization for deep learning.

A third element, quantum computing, can be introduced into this relationship. As seen above, optimization problems have large state spaces that need to be traversed almost exhaustively to find the solution. It has also been seen that this is one of the tasks in which quantum computation is superior to classical computation. Therefore, one can speak of a trichotomy or triad that relates the three fields and that, in the center, houses a field of algorithm research with the ability to have a high impact on certain problems.

Returning to the specific case of QML, it is known that a generic ML algorithm, whether classical or quantum, performs reasoning, usually inductive, on data. It takes concrete examples and tries to generalize to obtain rules common to all examples that allow it to make predictions or classifications. To do this, the algorithm, as it processes input examples, generates hypotheses that explain these examples. Some hypotheses will be more concrete, accurately covering certain cases but being wrong in many others, and, on the contrary, other hypotheses will be more general, covering most of the examples processed, but without accurately describing any of them. Thus, the set of hypotheses generated is called the hypothesis space, which is equivalent to the state space of an optimization problem. On this hypothesis space, it is necessary to look for the one that maximizes a certain value function that will represent a compromise between generality and abstraction as defined in each case.

This search, which is performed during the training process, is very costly for the algorithms. The greater the volume of examples processed and the greater the expressiveness of the system, the greater the number of hypotheses generated and the greater the similarity between them, the more difficult it will be to guide the search and the longer the training time.

In variational algorithms there is the same concept of searching the hypothesis space. The main difference is that the hypotheses are converted into parameter settings to vary the ansatz circuit. This process is performed by classical gradient descent algorithms, resulting in the barren plateaus (BPs) problem and, moreover, little generalization. It is evident that the same concept that works classically does not work quantumly, and therefore one must look for other models of both parameter configuration search and algorithms, going beyond the variational ones.

The QS&S algorithms are different from the gradient descent used so far. QS&S does not perform a search following the energy landscape and, therefore, cannot end up stuck in a plateau. The key is to leverage some domain knowledge in the form of heuristics to guide the search toward a solution. This heuristic must be run for each of the possible states of the problem being solved, something impossible for classical computation, but feasible for the quantum paradigm.

The key concept of QS&S for doing heuristic search is to include the computation of the heuristic as an additional operator to the set of operators in the search process. This will compute the heuristic value of the superposition of states, that is, the value of each of the states. Thus, it will be possible to increase the probability of the states with the highest reward by performing an exhaustive search. This concept of QS&S is not a concrete algorithm; it can be seen as a new philosophy, different from the current one, to develop QML algorithms. Later, this philosophy can be materialized in many types of algorithm, e.g., QMS.

By applying QMS to the N-Queen problem, a practical implementation following this QS&S philosophy for QAI has been demonstrated on a problem equivalent to the QAI problems.

5.2 Other QML techniques

Once the relationship between QS&S and AI and QML has been analyzed, it possible to explain briefly other techniques. This additional techniques study is interesting to see their problems because the QML field requires a different approach from the current one.

The field of QML is currently dominated by variational algorithms. These types of algorithms follow the hybrid paradigm with a quantum and a classical module. The quantum module is a circuit with parameterized quantum gates. Gate parameters are varied until the quantum circuit behaves in a certain way. To vary these parameters, a classical optimization algorithm is used. This algorithm takes the output of the circuit and evaluates it to determine the parameter values in the next run of the quantum circuit. To vary these parameters it follows a gradient descent based method. This system is run iteratively a certain number of times or until convergence is reached. An example of an architecture is in Figure 18. However, this paradigm of classic-quantum hybrid programming has not yielded good results.

This execution scheme closely resembles that of classical neural networks. There is an initial phase known as forward, where the network runs with specific neurons (equivalent to quantum gates) to produce a result (measurement of the circuit). Subsequently, a second phase of back-propagation occurs, where the parameters of the neurons are adjusted based on the output, following a gradient descent algorithm.

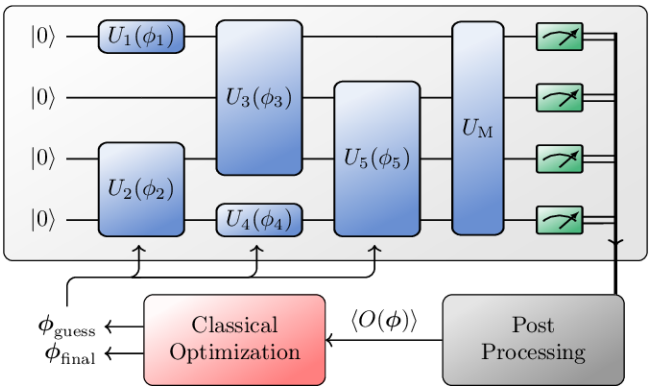

Figure 18: This figure shows a general architecture from a variational algorithm (extracted from [91]). The variational algorithms is composed by two main modules. The first module is a quantum circuit and it acts as a evaluation function. The parameters in the quantum module are modified following the optimization of a classical module, the optimizer. This process is repeated iteratively.

There has been much analysis of the reasons why variational algorithms do not work. The intuition was that variational algorithms would be more expressive than any classical algorithm and therefore learn faster and more accurately. On the contrary, so far the ansatz implementations have not been found to have more expressiveness than other classical algorithms [92] and it has been observed that the barren plateaus problem limits the trainability of the quantum circuit [93].

The barren plateaus (BPs) phenomenon occurs during the optimization process of the quantum circuit parameters of the variational algorithm. It is expected that each time the circuit is executed, the new parameter configuration of a function value that evaluates it will be closer or farther away from the solution. In this way, it is possible to know if the value of the evaluation function has improved or worsened with respect to the last parameter setting. In other words, the problem is expected to have a distribution in the evaluation function that allows guiding the search towards an absolute minimum. When looking at the BPs phenomenon, the distribution of the evaluation function value is flat in most of the space, except for a very small area around the solution that does have a downward gradient. This means that the search for parameter configurations of the quantum circuit cannot be guided, and it is necessary to carry out a practically exhaustive search, losing all possible quantum advantage [94]. An artistic conception, and very explicative, of the BP problem is shown in figure 19.

Reusing the most widespread metaphor for this situation, the ideal distribution is similar to a mountainous area, where it is easy to know whether the terrain is going up or down a mountain. In this ideal situation there are several mountains representing local maxima and one mountain higher than the others representing the absolute maximum. In this case, when

the peak of a mountain is reached, the local maximum, it is possible to see where there are other higher mountains and guide the search. On the contrary, the barren plateaus scenario is identified with a desert where everything is totally flat and it is almost impossible to guide the search and where there is a small well hidden in the sand where a local minimum is found.

There are multiple reasons for the existence of barren plateaus in problems, but the main ones are: circuit expressivity, local measurements, random initialization, and noise.

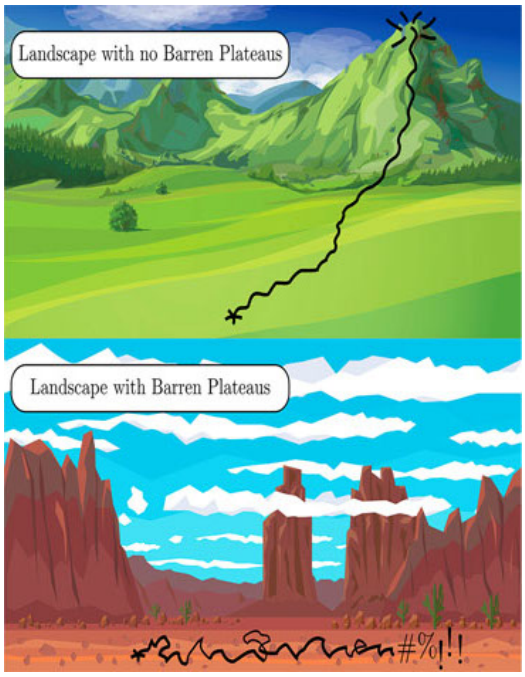

Figure 19: This figure shows an artistic explanation of phenomena of barren plateaus (extracted from [95]). The upper picture shows an ideal situation without Barren Plateaus, in which at any point is possible to decide which is the next step to be closer to highest point. The lower picture shows a situation similar to a desert in which is impossible to decide which direction should be taken to be closer to the maximum point.

After analyzing the BPs phenomenon in detail, some authors have come to the conclusion that variational algorithms are a dead end for the QML field [90]. This situation is problematic because much of the QML community effort was focused on this line.

After seeing the limitations of variational algorithms, the QML community has tried other fields in which to make relevant contributions. Once is the quantum backpropagation.

The classical backpropagation method allows adjusting the weights of a neural network to reduce the difference between its prediction of a data label and its actual label. This process is very expensive because the weights of all neurons in a network must be adjusted. In fact, it is the

process in which most of the training time of a neural network is spent. Because of its high cost and the possibility of parallelizing the process, something that is already classically done, the quantum computing community has tried to create quantum algorithms for backpropagation. However, this possibility has been discarded due to the need to execute non-linear functions and the impossibility of finding a quantum advantage [96].

The failed attempt to find a quantum algorithm in this field can be understood as further evidence that the community is still looking for a justification for research in the field of QML. It is also necessary to say that, once again, the attempt to find a quantum advantage is using a method based on gradient descent, similar to variational algorithms, and which has proven to be inefficient.

Another field that is being explored in great detail is that of quantum generative models. These algorithms are capable of generating data from noise. Generative artificial intelligence is a machine learning paradigm that differs from the usual discriminative models. The last ones are models used for classification or regression, while the former ones are models that aim to learn or discover relationships or patterns that exist between the different points in the input set, and then use this learned information to generate new points that mimic the characteristics of the input data set.

Quantum computing provides a perfect fit for this type of AI model for several reasons. First, quantum computing is inherently random, so it is able to generate samples from an ensemble stochastically without introducing any bias. Second, it allows one to compute gradients faster than its classical counterpart. And third, work has already been done that tests these algorithms on current quantum computers, proving that this is a viable solution.

Quantum generative models have been proposed as one of the strongest candidates to take advantage of quantum computing. To this end, metrics have been created that measure the generality of these models and compare them with classical models [97, 98]. Some examples of this quantum generative model are quantum GANs [99, 100] and parameterized quantum circuits [101] and its variations [102, 103]. In addition, small examples of these architectures have been implemented in quantum hardware [104]. There are also practical applications that make use of these quantum generative models, such as the discovery of new drugs [105]. An example of the architecture of a hybrid generative algorithm is shown in figure 20.

However, although they show great promise and are good candidates for quantum advantage, many of the current generative models are still based on gradient descent techniques, which causes them to have problems similar to those of other variational algorithms. Therefore, in the field of QML, these algorithms are beginning to be seen as useful in the medium to long term.

The limitations observed in the QML field are due to two main factors that are closely related

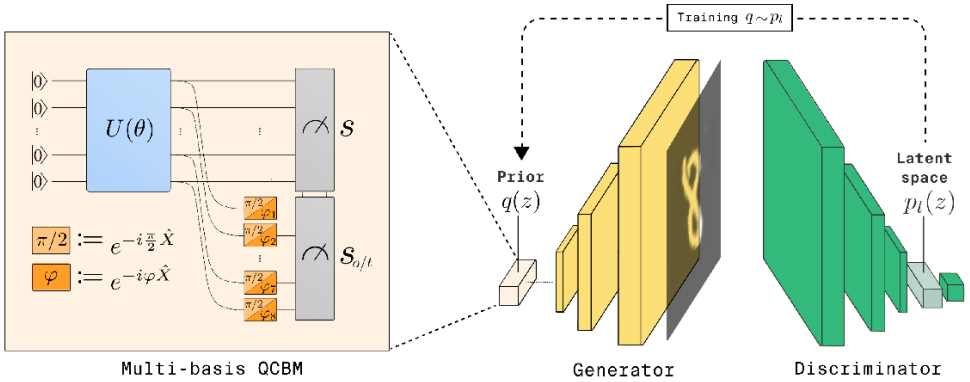

Figure 20: Architecture of a hybrid quantum-classical generative algorithm. It has a quantum prior module and the generator and discriminator are classical modules. (extracted from [104]). The quantum prior samples from noise, then, the classical generator and discriminator are trained to transform the sample into data.

to each other, the short field run and the imitation of classical techniques. The field of QML is barely less than 10 years old, the first papers on quantum ML algorithms are from the mid-2010s. Just as happened with brain mimicry as the starting point for classical ML, mimicking classical algorithms has been the starting point for QML. Therefore, the current algorithms are, practically, the first attempts of algorithms in this field.

However, it has been shown that this attempt to copy classical algorithms has serious limitations because gradient descent techniques are useful in classical algorithms but not in quantum algorithms. The future lies in designing new algorithms that, although based on the well-known concepts of classical ML, increasingly move away from the classical part and begin to exploit more purely quantum techniques.

In this thesis, it is proposed that the field of QML be brought closer to QS&S based techniques. This involves replacing gradient descent techniques with more informed searches. In classical ML, it is not possible to do an informed search in an algorithm like Neural Networks because it would take too much time to adjust the weight of each neuron. However, quantum computing can do exhaustive informed search with advantage over classical computing. The key to this advantage is that the evaluation function can be a quantum operator running on the superposition of states, [106] and that the search for the marked element is quadratically faster.

To demonstrate that the QS&S concepts seen in the previous sections can help, it is explained below how QMS has been applied to the N-Queen problem. This problem has been used many times as a benchmark for testing search algorithms that then have an impact on other ML techniques.

5.3 The N-Queen problem

The N-Queen problem consists of distributing n queens over a chessboard of dimension $n \times n$ without attacks between the queens, i.e., none of the queens coincide in the same vertical, horizontal or diagonal line. This is a well-known problem in the classical computing world because it is very easy to test different algorithms to find the solution.

In the Publication 4.3 of this thesis, this problem is used to test an algorithm that evaluates all combinations and keeps those that meet the constraints. The solution has been proposed using a heuristic that evaluates the quality of the solution based on the number of attacks between queens until a state is found in which the number of attacks is 0.

The N-Queen problem has been used as a benchmark for ML algorithms because the search necessary to find the solution is equivalent to the search that an ML algorithm performs in the hypothesis space to find the one that maximizes a certain function.

Thus, in order to prove that QMS works well with the N-Queen problem following the QS&S methodology explained above, it is necessary to have one more argument to continue this line of research.

To take the step of using QMS for QML, it is required to change the evaluation function of the N-Queen problem to the evaluation function needed in QML. For example, having a classifier that can discriminate the examples by 3 parameters, it is possible to create an evaluation function that determines the quality of a hypothesis according to the value of those three parameters. Then, it should apply that evaluation function to all possible states or configurations of the system until the maximum reward is found.

The results of applying QMS on N-Queen show that there is an advantage over classical algorithms. This consolidates the research line and prepares it for future steps focused on building the generalization process around QMS.

A final consideration regarding the N-Queen problem and QMS is the execution in quantum hardware. The results detailed in publication 4.3 were obtained in a simulator using logical qubits fully connected between them and without noise. However, as it was explained in previous chapters, quantum hardware still has some limitations to be able to execute circuits like a quantum simulator running in a classical hardware does. In a quantum hardware, the quality of these qubits is lower, so it is necessary to increase the number of qubits or reduce the problem. The number of qubits required for QMS is $PQ + \lceil \log_2 P \rceil + a + 2$, being P the number of parameters, Q the number of qubits for the discretization and a the number of qubits for the ancilla. Using two parameters with two qubits and three ancillas ($P = 2$, $Q = 2$ and $a = 3$), 10 qubits are required. It is a reduced version of the problem but it is useful to understand the

requirements. Using this configuration, around 1250 quantum gates are necessary.

5.4 Results

- ✓ QMS and the QS&S paradigm have been successfully applied to a problem equivalent to any QAI and QML problem, the N-Queen problem.
- ✓ A new approach to QML problems has been studied, the informed search in state or hypothesis space, as opposed to the mostly used uninformed search, e.g., classical gradient descent optimization.
- ✓ The alternative of using QMS offers the advantage of a search without the phenomenon known as barren plateaus.
- ✓ Scaling advantage results have been obtained using QMS which has a polynomial scalability in the number of qubits to represent the states.
- ✓ By applying QMS to the N-Queen problem and looking at its results, a tool has been proposed that can serve as a basis for other QML algorithms.

6

S&S applied to Space Exploration

The field of space exploration encompasses any field of research that involves learning more about any process outside our planet. Under the umbrella of the concept of space exploration, there is a heterogeneous group of research lines and techniques ranging from the design of new rockets to the analysis of earth data taken from satellites in low orbit. Within space exploration, there are more theoretical lines, such as models for predicting the movement of the planets, or more applied lines, such as communications satellites.

A feature common to many techniques of space exploration is the complexity of the analysis of data obtained by satellites or exploration missions. This complexity derives partly from the volume of data to be processed and partly from the complexity of the data. Of course, this complexity cannot be generalized to any line of research, but it tends to be a constant in many problems to be solved [107].

Examples of space exploration problems include analysis of Earth observation data, optimization of satellite orbits, prediction of solar storms, improvement of communications between satellites and ground stations, or analysis of complex data such as gravitational waves. These are just a few examples of space exploration use cases, but in all of them, quantum computing has been applied in some way or another [108, 109, 110].

In fact, the world's major space agencies are devoting great efforts to developing quantum technologies applied to space exploration. Some examples are NASA with its QuAIL and AMES groups, ESA with its ACT and PhiLab groups, JAXA with the Space Exploration Innovation Hub Center or the Chinese space agency with its Mozi and Micius satellites. But this is not limited to government agencies; around them there are large companies that have planned lines of research that apply quantum computing to space exploration problems to overcome the major limitations of classical algorithms.

In this use case, QMS can be applied to space exploration problems if they can be converted into optimization problems. In some cases, a conversion is not even necessary, as the problems are already studied as such as optimization problems. Specifically, this work will include as a use case the study of the inference of gravitational wave source parameters through an adaptation of QMS to the problem, known as *qBIRD*, quantum bayesian inference with renormalization and downsampling.

6.1 Introduction to Gravitational Waves

A gravitational wave is a perturbation of space-time produced by an accelerated massive body. They travel at the speed of light, contracting and stretching anything in their path. They represent ripples in the fabric of space-time, generated by temporal variations in the quadrupole momentum of the source mass, and induce variations in the length of the objects they pass through. This work is focused on gravitational waves produced by two black holes that rotate about themselves until a merge event occurs.

Gravitational waves are really difficult to measure and were theoretically proposed by Einstein, albeit with moments of disbelief and doubts encompassing a long and winding road. Today, gravitational waves produced in a merge event are measured because they are of such intensity that they can be measured because of the deformation of the detectors; the rest of the gravitational waves are invisible to the measuring devices currently available.

In 2015, the LIGO collaboration was able to make the first direct measurement of gravitational waves thanks to large laser interferometers. The detection of gravitational waves is an important new validation of the theory of general relativity.

The LIGO scientific collaboration is the union of different universities and research centers for the study and detection of gravitational waves. Members of the collaboration have access to the advanced LIGO detectors located in Hanford, Richland, and Livingston, Louisiana, both in the USA and to the VIRGO detector located in Pisa, Italy. This collaboration was initially involved in building the detectors with laser interferometer technology and is now involved in upgrading the interferometers and studying all aspects of gravitational waves. The founders of LIGO received the Nobel Prize in Physics for their work in detecting gravitational waves in 2017.

Studying gravitational waves in detail has a great impact on the study of the evolution of the universe and of phenomena that are still a mystery, such as black holes. A fundamental characteristic of the gravitational waves that are measured is that they have a validity period. This validity period is a problem because the volume of detected events is constantly increasing due to improvements in detector accuracy. If the usual trend of slow processing rate and high

detection rate continues, a point of collapse will be reached. In addition, once the phenomenon disappears, it is not possible to observe it, the only possibility is to analyze the static and past already recorded data.

Gravitational waves make possible the opening of a new window through which to look at the universe, complementary to the observation of the electromagnetic spectrum. This new method promises the observation of astrophysical events that have never been seen before.

One of the possible analyses that can be done on gravitational waves is to obtain some of the parameters of the event that generated them. It is possible to obtain the distance they were from, the masses of the objects, their spins, etc. This analysis is done with one of the techniques with more results as parameter estimation, giving rise to the field of parameter estimation gravitational waves (PEGW).

Currently, the PEGW analysis process is slow and requires a high processing capacity and computational power. This is because it requires analyzing many possible states and combinations of variables. PEGW can be viewed as an optimization problem in which the value of certain parameters must be adjusted to find a synthetic distribution that is as close as possible to the actual distribution.

In order to better understand PEGW, it is necessary to explain the analysis process. The interferometer that detects a gravitational wave is composed of two large lasers. These lasers are joined at a perpendicular intersection and when they detect a gravitational wave, they generate an interference pattern, as shown in figure 21. This pattern contains all the information about the event that generated the gravitational wave. Therefore, if, by adjusting the value of certain parameters, it is possible to create a sufficiently similar numerical wave, the value of the parameters that form or create that interference pattern can be found. In this way, a state space is created with a number of dimensions equivalent to the number of parameters in which it is necessary to try all possible combinations to find which one generates a synthetic interference pattern as similar as possible to the measured one, as explained in Figure 22.

Following this PEGW scheme and in order to analyze in detail the gravitational waves as a whole, both the LIGO collaboration and other universities have implemented different gravitational wave analysis software. Among them, two stand out, PyCBC [113] and Bilby [114]. Both allow inference, plotting, etc. In this work, the aim is to create a hybrid algorithm that includes quantum computation modules for PEGW and that can be included with some of the existing tools or even be a tool in itself, in order to speed up the PEGW process.

The classical PEGW algorithms are based on the classical Metropolis-Hastings algorithm seen in the chapter 2. These algorithms encounter computational constraints similar to those experienced by classical optimization algorithms. In addition, in this case, an evaluation func-

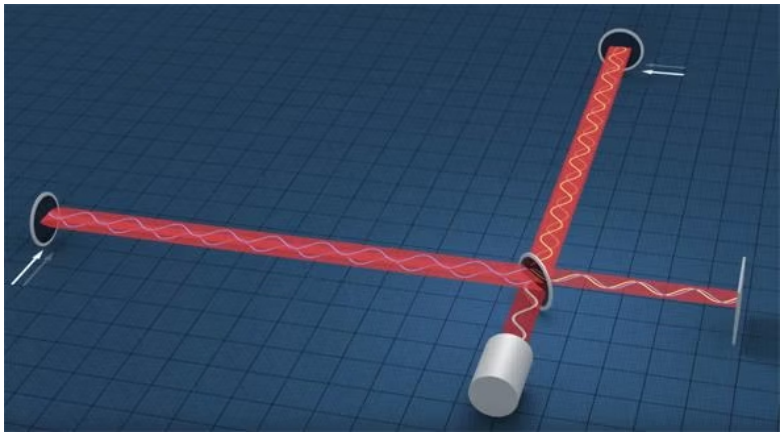

Figure 21: Scheme extracted from [111]. It shows how the interferometers built by LIGO works. Both interferometers are in a perpendicular position. If a gravitational wave event is detected, a interference parameter is received in the detector.

tion, likelihood, slow to run, and a large volume of data are added. All this makes the number of gravitational wave events to be analyzed grow faster than their analysis capacity, leading to different scenarios of collapse of the gravitational wave analysis [115].

The ultimate goal of introducing quantum computing into the field of gravitational waves is to create a pipeline that fully utilizes quantum technologies, both in detection and in processing and analysis. However, this goal is too general and ambitious. Therefore, it is necessary to go step by step, and creating a quantum PEGW algorithm that gains an advantage over classical algorithms is already a breakthrough.

To carry out the development of a PEGW algorithm with quantum computing, after understanding the problem as an optimization, QMS has been adapted to the problem with certain variations. This new version has been called qBIRD.

6.2 Methodology

Understanding the PEGW problem as an optimization problem in which a search is required, it is necessary to define an evaluation function that determines the quality of each state. In this case, loglikelihood is used as the evaluation function.

The likelihood function is the conditional probability of the Bayesian theorem $\boldsymbol{\mathcal{L}}(d|\boldsymbol{\theta}) = p(d|\boldsymbol{\theta})$. Being d the data observed by the interferometer, θ the set of parameters to infer that explains d, $\boldsymbol{\mathcal{L}}$ the likelihood function and p the posteriors of the data and parameters. For this

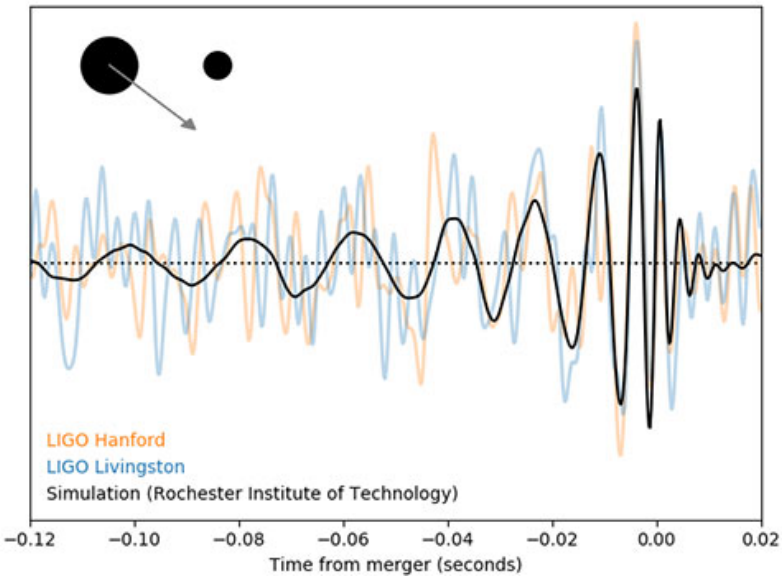

Figure 22: Scheme extracted from [112]. It shows the gravitational waves detected by LIGO Handford (yellow) and LIGO Livingston (blue) and the simulation of the parameter estimation process (black). The estimation process tries to adjust the inferred wave as much as possible to the received wave.

purpose, it is common in the literature of gravitational wave astronomy to assume a Gaussian noise in the detectors [116], so that the Gaussian-noise likelihood function has the form:

$$\mathcal{L}(d|\boldsymbol{\theta}) \propto \exp\left(-\frac{1}{2}\int_0^\infty \frac{|\tilde{d}(f) - \tilde{\mu}(\boldsymbol{\theta}, f)|^2}{\sigma^2(f)} df\right), \tag{19}$$

All these functions are computed in the Fourier domain, where f represents frequencies. Here, $\tilde{d}$ denotes the data set acquired conventionally by the detector, $\tilde{\mu}$ signifies the waveform derived from a specific relativistic numerical model calculated using the parameter set θ, and σ indicates the detector noise, expressed as spectral density. Notably, although the integral spans all possible frequencies, in practice, this interval is confined to frequencies compatible with ground-based detectors while excluding certain frequencies like harmonics of domestic electricity. This restriction ensures that σ remains non-zero across these frequencies, thereby preventing any divergence in equation 19.

Classically, the M-H algorithm is used to perform PEGW inference. The idea is to run the M-H algorithm many times; it can be run up to 40,000 times. In each run, the M-H algorithm will traverse the state space and stop at a point. All of these points are then taken together to perform a statistical study of the most probable set of values.

The concept of *qBIRD* is to use a quantum M-H algorithm, seeking greater precision in each execution, and therefore to perform a smaller number of executions. In addition, *qBIRD* is designed so that, in the future, with a more developed quantum HW, the likelihood calculation will be an additional operator in the quantum circuit, which makes the calculation of each iteration faster.

Due to the limitations of current quantum hardware, the likelihood calculation of each state must be classically preprocessed and introduced into the quantum circuit via a QRAM. However, some authors have already proposed the calculation of quantum likelihood [106]. By including the likelihood calculation as an additional operator, it can be applied over the entire overlay and saves both time and resources by avoiding the inclusion of the QRAM technique, which adds a lot of depth to the circuit.

When comparing the classical PEGW algorithm with its quantum version, it is necessary to use some figure of merit. Since *qBIRD* is an adaptation of QMS, it is straightforward to use the TTS explained in subsection 4.2. Analyzing the TTS of classical and quantum PEGW methods shows that there is an even greater advantage over the use of other PEGW use cases. Moreover, this advantage is expected to be even higher when larger problems can be run.

Another advantage of *qBIRD* over purely classical algorithms is its scalability in qubits inherited from QMS. *qBIRD* uses exactly the same problem representation as QMS and scales polynomially with the number of parameters being analyzed.

Currently, quantum simulators running on classical hardware can simulate up to 4 parameters with a discretization of 5 or 6 qubits each. For this, the rest of the parameters are fixed to complete the 17 that are usually inferred. As the capabilities of quantum computers increase, this number of inferred parameters will increase to 17 and be an inference equivalent to the state-of-the-art.

Different tools are used to represent the results obtained in the PEGW process. The main one to be seen in this work is the corner plot, shown in figure 23.

A corner plot, like the shown in figure 23, is a commonly used graphical representation in parameter estimation in the field of gravitational waves. This type of plot effectively visualizes the joint distribution of multiple parameters of the model being analyzed. Each axis of the plot represents a pair of parameters under investigation. Along the margins of a corner plot, histograms are included showing the marginal distribution of each parameter individually. The points or contours in the plot represent samples drawn from a Markov Chain Monte Carlo (MCMC) chain or another algorithm used to explore the parameter space of the model. Summary statistics such as the mean or median of the distribution of each parameter and confidence intervals are included to highlight important features of the distribution.

Corner plots are useful tools for understanding the relationships between multiple parameters in a model and for identifying possible correlations between them.and it is especially interesting when there are few parameters, 2 or 4, as in this case, because it allows to see the posterior probability distributions (PDFs) of the parameters 2 to 2.

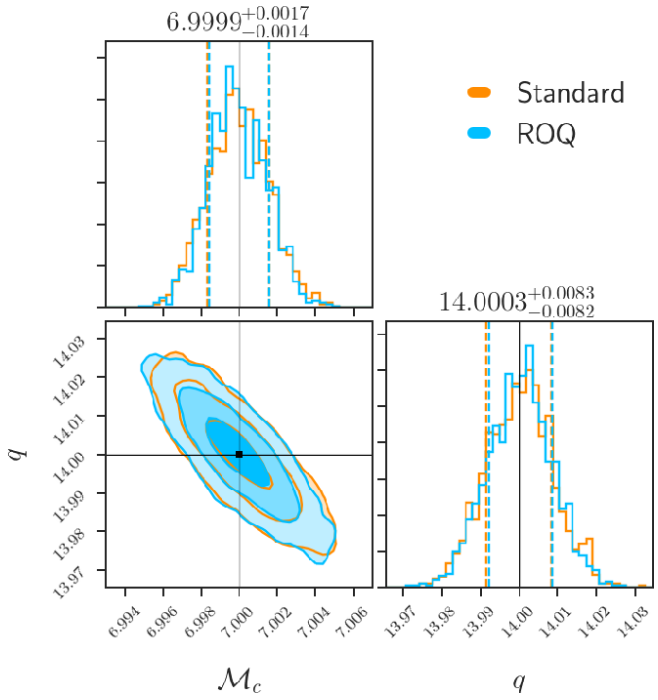

Figure 23: Corner plot extracted from [115]. It shows a combination of the result of estimating two parameters of a gravitational waves. It is a general format to show the results in PEGW.

As discussed above, the aim of *qBIRD* is to develop a PEGW algorithm that will speed up this process and overcome current bottlenecks. *qBIRD* may end up being included in one of the currently discussed libraries, PyCBC or Bilby, or even being a new library in its own right. To this end, in this work, there has been a close collaboration with a scientist from the LIGO collaboration.

6.3 qBIRD

The simplest way to understand *qBIRD* is as an optimization algorithm capable of obtaining good results on the PEGW problem. This algorithm uses sample and search techniques to find the region of the state space in which the likelihood value is highest. It then samples that region

in order to find the probability distributions of the posteriors. The search is guided using the likelihood function explained in equation 19 with the objetive of maximizing it.

The QS&S technique used by *qBIRD* is an adaptation of the one used in QMS, with the objective of adjusting it to the PEGW problem. For this purpose, the concepts of renormalization and downsampling are used.

Renormalization is a technique represented by a semigroup of transformations that make up an iterative process of elimination (called truncation) of degrees of freedom of a quantum system according to a certain criterion in order to obtain an effective description of the retained degrees of freedom [117]. The criterion can be of low energy, maximum likelihood, etc. It gives rise to approximate computational methods of variational type. This technique can be applied to QS&S.

QS&S techniques are primarily required for exploring large state spaces. When dealing with quantum computing in the context of a significant state space, the challenge lies in the probability distribution. In smaller state spaces, the probability of finding states with lower energy functions is relatively high compared to other states. For instance, in a search involving 8 states, the normalized probability of lower energy states might be around 0.3, contrasting with other states at 0.001. However, in a state space of 2^{32} states, the maximum probability state could still be 0.001, with the next maximum probability state potentially as low as 0.000001. Despite the three-order difference in magnitude, both states become indistinguishable upon circuit measurement. Therefore, it becomes necessary to execute iteratively the quantum circuit, eliminating the least probable states in each iteration until a noticeable difference in probability is achieved after measurement. This iterative process is based on renormalization.

This renormalization-inspired technique that is an adaptation of the QS&S to PEGW problem, called qBIRD, can be seen from a different perspective. In computer science, specially in the algorithms of digital signal processing, there is a set of algorithms known as downsampling techniques. These techniques are also widely applied to convolutional neural networks [118].

Downsampling refers to the comprehensive process of reducing both bandwidth (filtering) and sample rate. When applied to a sequence of signal samples or a continuous function, downsampling yields an approximation of the sequence that would have been obtained by sampling the signal at a lower rate (or density, as seen in photograph resolution reduction). In the context of qBIRD, downsampling is employed similarly to approximate or reduce the state space to get the likelihood distribution that requires sampling.

qBIRD receives as an input a set of parameters to infer and it returns the posterior values of these parameters according to a likelihood function. These parameters are initialized to a random values according to some prior knowledge. The prior is composed by an interval and

the distribution of this interval. If the distribution is unknown, a Gaussian function is used. The first step of the algorithm is the proposal of new values, *Start* in Figure 29. Then, steps 1 to 3 are executed.

Step 0. Parameter initialization: The algorithm begins by initializing values for each parameter drawn from a prior distribution, which is defined by the lower and upper bounds of the prior function. These values are then stored in the state register.

Step 1. Renormalization & Downsampling: The first module implements the quantum Metropolis-Hastings algorithm adapted from publication 4.3, and incorporates a renormalization technique defining the *qBIRD* algorithm. During this phase, the quantum walk is iteratively applied to narrow down the parameter space, aiming to identify the values for each parameter that maximize the likelihood.

a) Quantum Metropolis: Iteratively apply the walk operator W, defined in Equation 17, a number of times on the initial state:

$$|\psi\rangle := W_L ... W_2 W_1 \left|\phi^{(0)}\right\rangle . \tag{20}$$

b) Sampling: Sample the state register.

c) Threshold condition: If the number of elements is the same than the number of parameters to infer, each parameter is represented by 1 qubit, jump to Step 2, otherwise calculate the number of elements.

d) Qubit reductor: Reduce the number of qubits in the state register by 1 unit and go to Step 1a). This condition allows to eliminate one qubit for each parameter, ending up with a minimum of one qubit per parameter.

The genesis of this second module stems from the challenge of employing the quantum Metropolis algorithm to pinpoint the state with the highest probability. Drawing inspiration from the renormalization techniques of quantum lattice models [117], it is addressed the issue arising from the vastness of the state space. In quantum systems, the normalization factor leads to minute probability discrepancies between the most and least likely states. Despite spanning several orders of magnitude, these disparities fail to attain significance without an impractical number of measurements.

By progressively eliminating states with significantly lower probabilities through the reduction of problem size and qubits, these discrepancies become increasingly discernible. Employing this technique enables the identification of the state with the maximum likelihood among all proposed values.

Step 2. Mean & Std Deviation calculator: The third module consists of a classical processing that takes the results obtained in the first two modules to generate PDFs for each parameter.

Step 3. Search interval calculator: To gradually narrow down the search area, a new interval for each parameter is proposed.

End. After a given number of iterations of Steps $0-3$, PDFs of each of the parameters are constructed from the values obtained in each iteration. This algorithm is vastly detailed in [3].

The strength of *qBIRD* is its ability to sample a distribution with a low number of iterations. As it is possible to observe in figure 24, using a different number of W it is possible to perform almost a perfect match with the distribution to guess (black line). While the classical sampler would need almost an exhaustive sampling, the quantum algorithm can do with a number of W between 3 and 5. It is just a small example, but very illustrative.

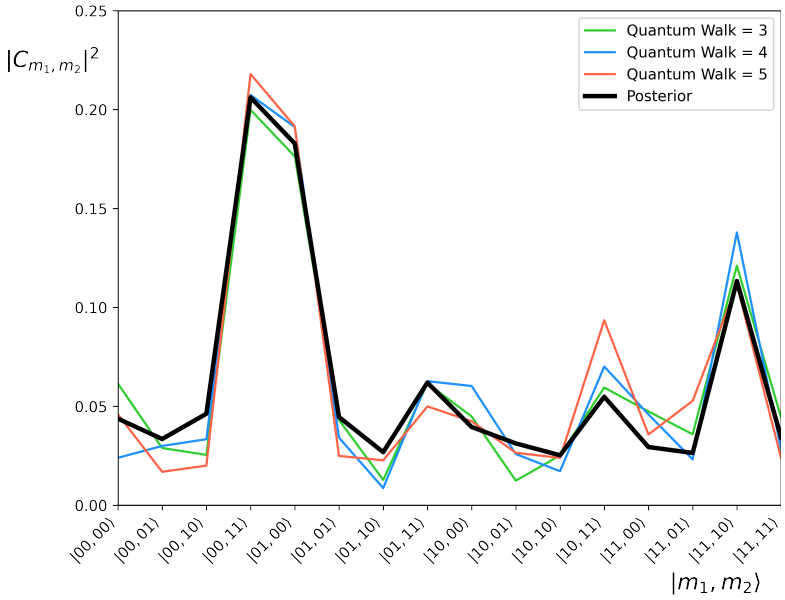

Figure 24: Colour lines represent the state register probabilities for different number of applications of the quantum walk operator. It is a two source masses inference, using $Q = 2$ discretization qubits, for the event GW150914. X-axis is composed by the 16 combinations that represents the binary encoding of a value for the first mass and another value for the second mass. The black line represents the posterior probability function of each of these combinations.

Once *qBIRD* was designed, it was implemented in a circuit. Subsequently, this circuit was executed in a quantum simulator, which ran on a classical computer utilizing the Qiskit library

[119]. The results obtained from the simulator are invaluable, as they validate the functionality of the algorithm. The next step, executing *qBIRD* on quantum hardware, does not require any special implementation. This implies that qBIRD's limitations are solely dictated by the constraints of quantum hardware, but it can scale as much as needed.

qBIRD has been tested using two benchmarks, speed and accuracy. Both metrics are relevant for any quantum algorithm called to replace a classical one. As explained above, the role of *qBIRD* is to replace the classical samplers integrated in tools such as *Bilby* and *PyCBC*. *qBIRD* must be fast because current PEGW algorithms take unacceptable times for the speed at which events are detected. In addition, its results must be as accurate as or more accurate than classical algorithms to be accepted by the gravitational wave community.

The core of *qBIRD* is the quantum Metropolis-Hastings algorithm based on equation 17. To test the execution time of *qBIRD* applied to PEGW, it is necessary to prove the performance of W applied to the problem. In the work [2] is explained how the quantum Metropolis-Hastings algorithm is able to get a scaling quantum advantage as shown in figures 25 and 26. It is important to notice that in both figures, the results are obtained using real LIGO data.

In both figures, TTS metric is used compare classical and quantum algorithms. As it has been shown in other results, the simulator limitations do not allow to execute problems with a size equivalent to the solved by classical computers. However, it is possible to analyze the scaling tendency, and the quantum advantage is appreciable. In the table 1 a summary of the scaling exponent is shown in each inference case.

This table 1 presents the same information as figures 25 and 26 in a more concise format. The key information conveyed by the figures is the value of the exponent, where a value lower than 1 indicates quantum advantage. Therefore, a lower exponent signifies better quantum advantage, with 0.5 being the maximum due to the quadratic advantage defined by Grover. Additionally, it is observed that the exponent improves, approaching 0.5, with an increasing number of parameters to infer. This observation is linked with figure 16 and the problem size: with two parameters to infer, the problem is considered small, whereas with four parameters, it falls closer to the medium-sized category, where the gap between classical and quantum performance diminishes.

2 PARAMETERS INFERENCE

Parameters inferred	**Fit Exponents**
Chirp mass and mass ratio	0.95
Dimensionless spin_1 and spin_2	0.89
Comoving volume and inclination	0.87
Comoving Volume and coalesence phase	0.88

4 PARAMETERS INFERENCE

Parameters inferred	**Fit Exponents**
Coalesence time, mases and comoving volume	0.68
Dimensionless spin_1, spin_2, comoving volume and inclination	0.67

Table 1: Fitted exponents for different sets of variable parameters.

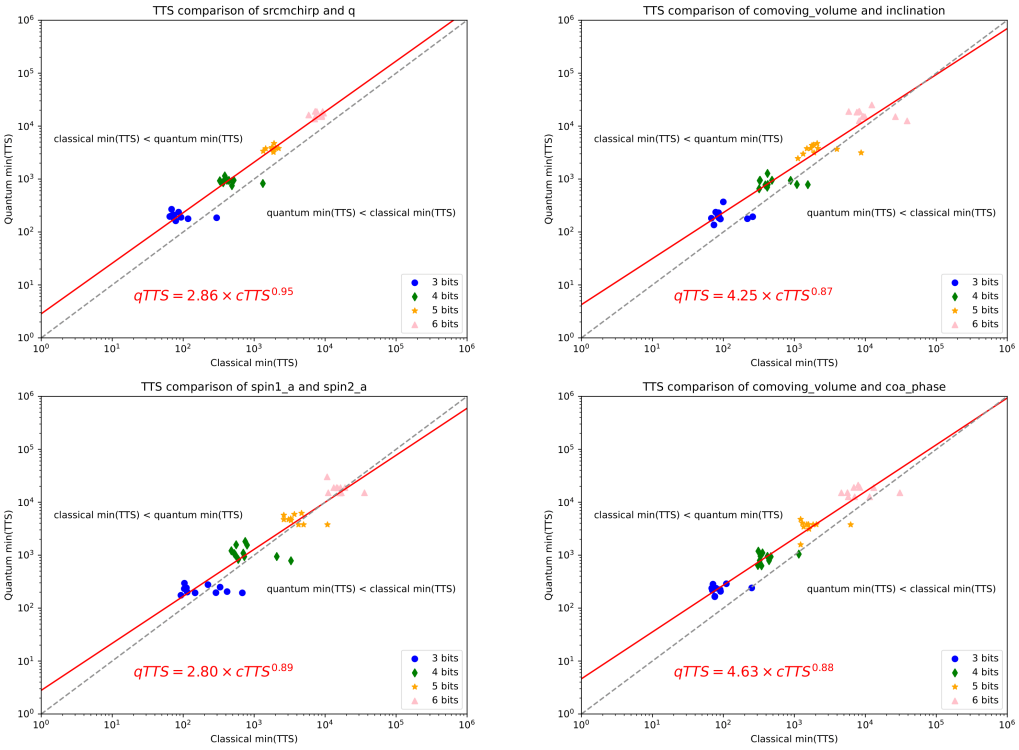

Figure 25: Comparison of the minimum value of the classical and quantum TTS achieved for the 2-parameter inference simulations. Top left the source-frame chirp mass and the mass ratio have been inferred. Top right the comoving volume and the inclination have been inferred. Below left the dimensionless spin-magnitude of the larger and smaller object have been inferred. Below right the comoving volume and the coalescence phase have been inferred. All results have been obtained for 3-6 bits of precision with a constant value of β.

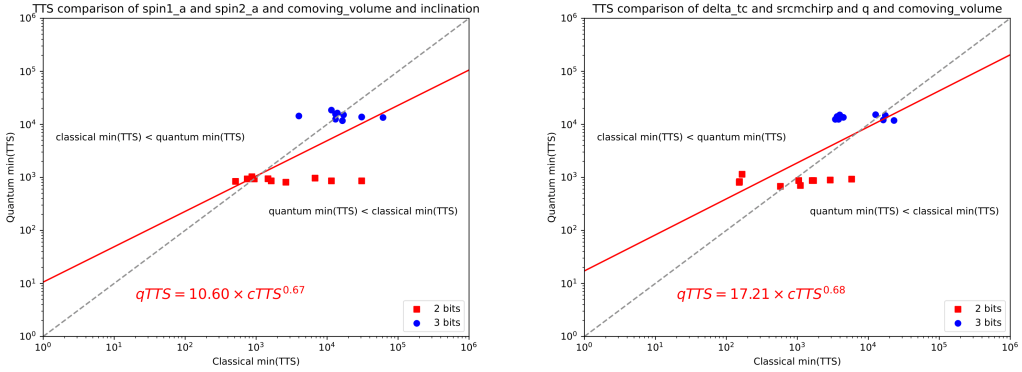

Figure 26: Comparison of the minimum value of the classical and quantum TTS achieved for the 4-parameter inference simulations. On the left the dimensionless spin-magnitude of the larger and smaller object, the comoving volume and the inclination have been inferred. On the right the source-frame chirp mass, the mass ratio, the coalescence time and the comoving volume have been inferred. All results have been obtained for 2-3 bits of precision with a constant value of β.

Just as important as the speed of execution is the precision and, derived from this, the visualization of the data. This is the motivation of the second publication that supports this work [3]. In this continuation of the first work, qBIRD is used to get corner plots with real data similar to those used by GWs community.

In the figures 27 and 28 it is shown the results of building corner plots using *qBIRD*. Both cases are generated using injected parameters. The injection option is very used in the GWs community to test the precision of the algorithm. In both figures, the injection value (an orange cross) is the middle of the inference, falling always in the dark blue region, the region marked by *qBIRD* as the more likelihood or the more posterior values region. Both corner plots are tests of the accuracy of the algorithm.

Finally, if the algorithm has demonstrated an advantage in execution speed and good accuracy, the final test to ensure that the quantum algorithm can outperform a classical one, once the quantum hardware limitations are solved, is to show its scalability. It is possible to prove that the scalability of *qBIRD* is polynomial in the number of parameters, $\mathcal{O}(P)$. The point is to analyze the qubits required by the quantum walk.

The quantum walk employs a total of $PQ+\lceil \log_2 P \rceil + a + 2$ qubits: PQ represents the product of P, the number of inferred parameters, and Q, the number of discretization qubits, with 2^Q states represented for each parameter; $\lceil \log_2 P \rceil$ qubits to represent register of each parameter in binary encoding; a qubits to represent the auxiliary register for the acceptance probability. Finally, 2 qubits are needed, one for the movement register, and another for the coin register to

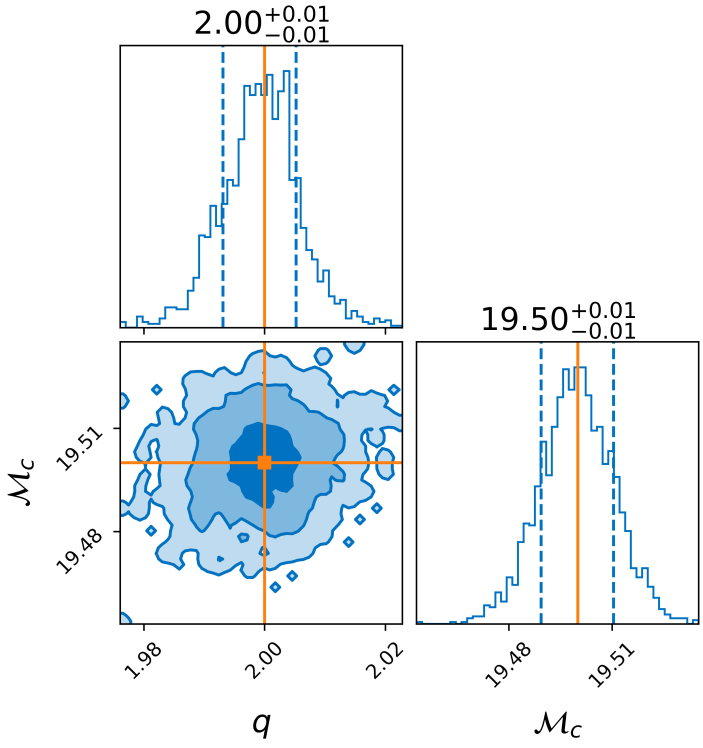

Figure 27: Posterior distributions obtained with *qBIRD* for the chirp mass $\mathcal{M}_c$ and mass ratio q of a simulated BBH gravitational wave signal injected into Gaussian-noise using *PyCBC*. The injected values are $\mathcal{M}_c = 19.50$ $M_\odot$ and $q = 2.00$, shown in orange.

encode the accept/reject probability of all states.

All terms in the scalabilty formula are dominated by the number of paramters and it results in a good scalibilty, which has a lower consumption in the number of qubits than the qubit growth trend of quantum hardware.

The execution in quantum hardware following these number and using two paramters with three qubits of discretization and three qubits of ancilla will result in a 12 qubit execution. If qBIRD uses four W operators at each repetition, the number of gates will be close to 6.000 gates including H, X, CX, CCX, MCX, CU and RY gates.

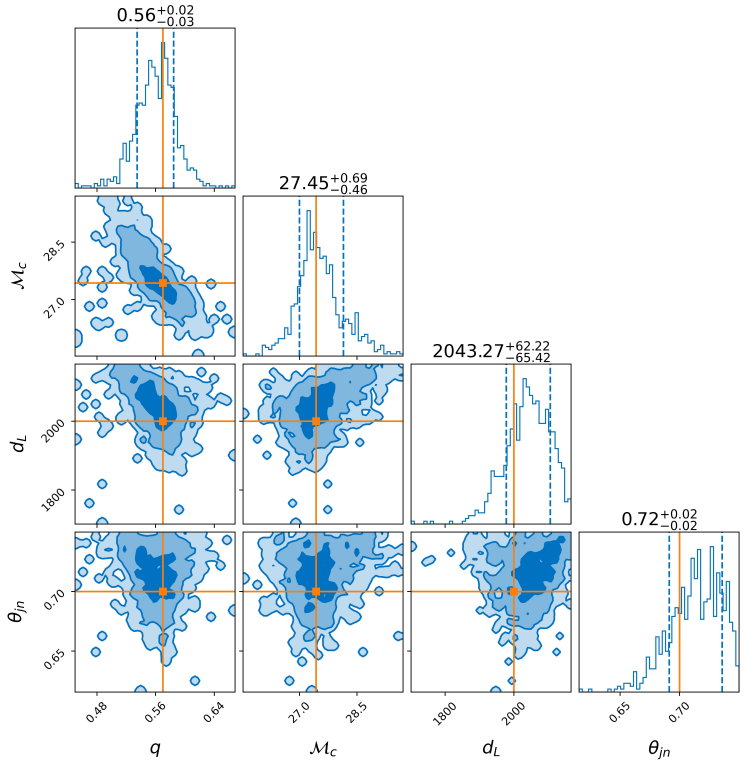

Figure 28: Posterior distributions obtained with *qBIRD* for a synthetic BBH signal characterized with 4 unknown parameters. The simulated gravitational wave was injected into zero-noise using *Bilby*, and the injected values are $\mathcal{M}_c = 27.43$ $M_\odot$, $q = 0.57$, $d_L = 2000$ Mpc, and $\theta_{jn} = 0.70$, shown in orange.

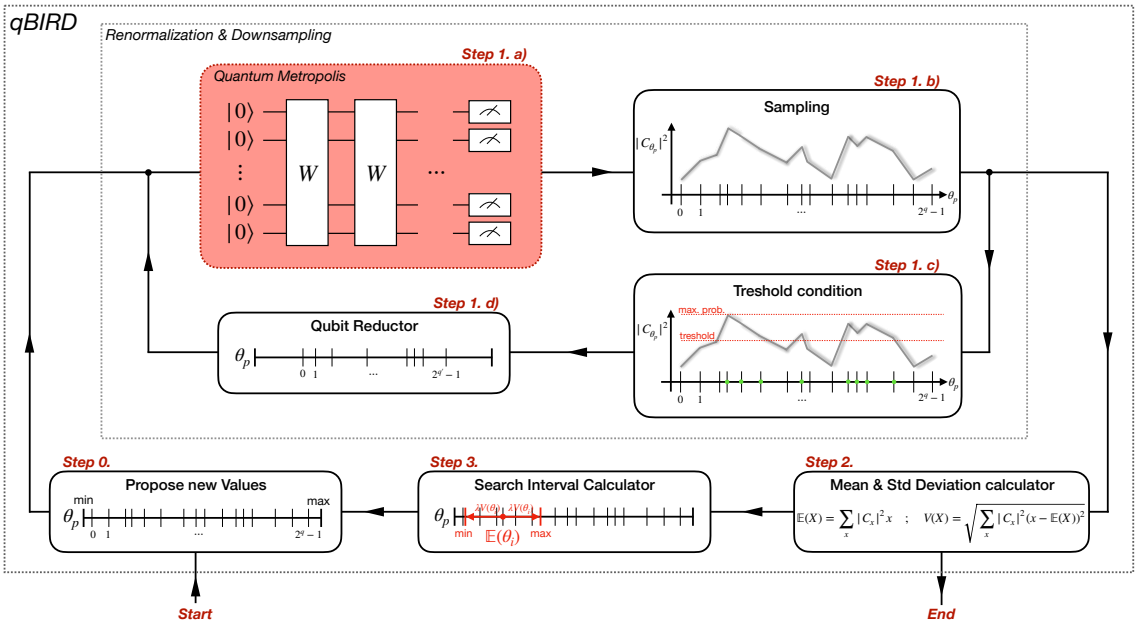

Figure 29: Flowchart of *qBIRD* algorithm. This algorithm is divided in three steps. Step 1 is focus on renormalization process. This renormalization executes the quantum metropolis circuit and reduce the search. Then, steps 2 and 3 calculates the new intervals for the variables and the process starts again.

6.4 Conclusions

- ✓ QMS has been correctly adapted to a new use case in [2] and [3], the inference of gravitational wave parameters, thus demonstrating the high versatility of QMS as a generic tool to adapt to other optimization problems.
- ✓ Quantum advantage results have been obtained using data from real gravitational wave events in [2] and [3].
- ✓ Results have been obtained with an accuracy similar to that of state-of-the-art methods using real data from gravitational wave events in [2] and [3].
- ✓ Generation of corner plots with accuracy similar to those generated by state-of-the-art techniques and using data from real gravitational wave events.
- ✓ *qBIRD* has good scalability in the number of qubits and better than the state of the art, which is essential for estimating 17 parameters.
- ✓ Modular software tool to make any module classical or quantum or to integrate new functionalities. In this case, the classical module that can be quantized is the calculation of likelihoods.
- ✓ Viable solution and new research line based on quantum software for an open problem in LIGO collaborations.

7

S&S applied to Protein Folding

The research field of quantum chemistry includes those processes of chemistry in which quantum phenomena occur. The cases treated by quantum chemistry must be modeled in order to study them and predict the processes that will occur. For example, to predict the behavior of a molecule under certain conditions, it is necessary to create a model that explains the interactions of the molecule with the environment.

The main limitations of the classical algorithms are to represent the superposition and entanglement effects between particles. The only solution for these cases is to create very simplified models that, although they do not predict accurately, it allows to know some characteristics of the systems. Although this is a very limited approach, it was the only possibility that existed until the birth of quantum computing.

The advantage of quantum computing applied to quantum chemistry models is that it can inherently represent many of the processes that occur without having to use extra storage and processing resources. Examples include calculating the minimum energy of a molecule, calculating the intermediate states in a chemical reaction, predicting the best structure of a system to have certain properties, or simulating the behavior of complex molecules in a certain environment, for example, the protein folding problem.

Quantum chemistry is fundamental in the development of both industrial processes and research. A large part of the research lines for the development of new drugs, new materials, carbon footprint reduction, etc. is directly related to quantum chemistry processes.

The fact that classical computation is very limited in this field of research and that the basis of quantum computation is perfectly adapted to the processes to be modeled, makes quantum chemistry the most promising candidate for quantum advantage in the short term. This means

that the first practical application of quantum computing is expected to be in a problem that, at some point, involves a quantum chemistry process.

This chapter has been approached as a use case of QMS and, therefore, of the application of the QS&S philosophy to different problems. The interesting quantum chemistry problems in this chapter are those that can be understood as optimization problems. Within quantum chemistry, there are problems that can be seen as a set of variables that can take different values and it is necessary to adapt these variables to find a solution. Specifically, the central problem that has been chosen is the protein folding problem.

7.1 Introduction to Protein Folding problem

The problem of protein folding is to find the three-dimensional structure of an amino acid chain that is stable and functional in the human body. For this to happen, it is necessary for the amino acid chain to adopt the three-dimensional structure of minimum energy. Naturally, the human body generates proteins folded in this minimum energy form, but introducing artificially a protein into the body, it is necessary to calculate its structure in such a way that it fulfills its task perfectly.

In the human body, there are many functions that are performed by a protein that attaches to a particular tissue and captures other substances. This is the case with insulin and glucose. Insulin is a protein capable of capturing free glucose in the bloodstream and transporting it to a storage point for metabolization. If the body cannot generate insulin due to a pathology known as diabetes, hyperglycemia occurs, which can be very harmful to the human body. To solve this problem, insulin proteins have been artificially designed, which when injected into the blood perform the same function as if they had been generated by the human body.

Research in the field has revelled the effects of diabetes by folding the amino acids that make up this protein so that it could perform its function in the human body. This structure is relatively simple because insulin is a small protein with few amino acids, but in other cases, proteins are much more complex, and the calculation of its structure can only be done by an algorithm.

Classical protein folding algorithms must calculate, for each pair of amino acids, all possible ways of binding. This extended to all the amino acids of a protein, which can be more than 100, scales exponentially and is uncomputable for a classical computer, at least with exhaustive methods.

Protein folding is the basis for the development of new drugs. In the human body there are only 20 amino acids that combine to create proteins. The functions of each of these amino

acids are well known. Therefore, and simplifying the problem, if a protein to perform a specific function is required, it is only necessary to create it using certain amino acids. The next complicated step is to go from a linear chain of amino acids to a three-dimensional structure. Once the protein structure is calculated, the response of the human body to certain stimuli can be tested, measure the degradation of cells after exposure to certain products, and so on.

In 2021, a historic milestone in the development of algorithms applied to protein folding occurred. The company Deep Mind released AlphaFold, a tool that uses deep learning technology to predict the structure of proteins given a chain of amino acids. AlphaFold proved to work very well for proteins of large complexity and size, which earned it the recognition of the entire community. While it has some limitations, such as training time or solution quality for some proteins with several possible structures, it is a breakthrough in the field.

AlphaFold makes a prediction based on convolutional neural networks equivalent to those used for image recognition in computer vision. This type of network does not take into account the energy of the predicted protein, therefore, the solution offered by AlphaFold or any other DL-based algorithm can be refined to find a structure of lower energy and, therefore, more stable. That is the motivation of using protein folding as a use case for QMS, to create a search algorithm that refines the solution given by a method that makes an initial guess. In this case, QMS has been adapted to protein folding in a quantum classical hybrid algorithm called QFold.

QFold has a hybrid architecture, apart from the internal architecture of QMS itself, in which a classical DL algorithm is run first, which gives a first structure of the protein that is not very stable, but serves as a starting point for the classical algorithm to search for the three-dimensional structure of the protein with minimum energy and maximum stability.

7.2 Methodology

To define the overall structure of the protein, the pairwise bonds of the amino acids can be defined. Each amino acid bond can be defined as two torsion angles, ϕ and ψ. The torsion angles, also called dihedral, are angles between the atoms in the backbone structure of the protein, that determine its folding. An example with the smallest of the dipeptides, the glycylglycine, can be found in Figure 30. These angles are usually three per amino acid, ϕ, ψ and ω, but the latter is almost always fixed at value π and for that reason, not commonly taken into account.

By discretizing the torsion angles between each of the amino acid pairs, the overall structure of the protein can be described. Then, all possible angle values of the protein are evaluated, calculating for each possible configuration its energy. In this way, the configuration with the lowest value in the evaluation function can be obtained following a search guided by energy minimization.

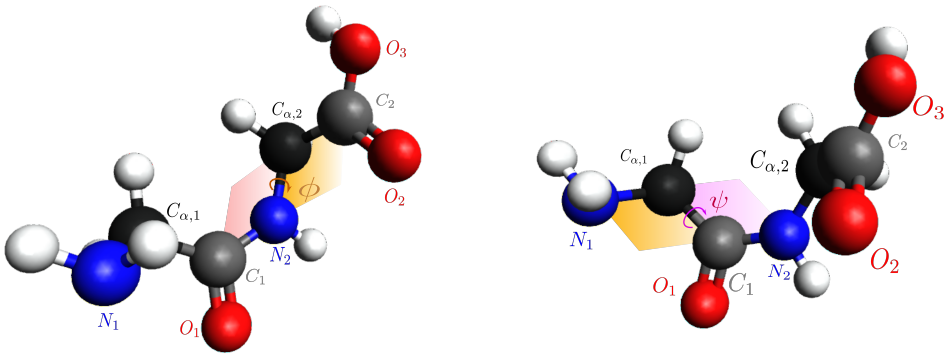

Figure 30: It shows the dihedral angles ϕ (yellow) and ψ (pink) of protein. Both angles are key to represent the bond between aminoacids. These angles are discretized with a number of qubits to calculate the 3D structure of the protein.

The protein folding problem has been extensively studied both from the perspective of classical and quantum algorithms. The latter, due to the limitations of the algorithms and their executability, have had to resort to very simplified models of the proteins. In fact, all models prior to QFold, use lattice representations for proteins. This means that the angles between amino acids can only be represented at right angles, 4 possible values. This representation is totally unrealistic and greatly limits the results obtained.

One of the main innovations of QFold, along with the results obtained, is the totally realistic representation of proteins. QFold represents both phi and psi angles and does so at any value between 0 and 360, namely between $-\pi$ and π, with a user-defined discretization. This discretization is limited by the runtime capability of the devices, but can be as large or small as chosen, as the performance of QFold is maintained. Obviously, using a discretization with a larger number of states, better is the performance of the quantum algorithm versus the classical one because it increases the complexity of the problem.

The main objective of QFold in the short term is to be able to refine the search done by another module. In the medium to long term, QFold can be considered as a stand-alone tool, but currently, classical algorithms have greater capabilities. Following this objective, the QFold architecture has two main modules, initializer and search algorithm.

The initializer is a classic deep learning module. Right now it is a module that uses a simplified AlphaFold algorithm called MiniFold. However, this initializer could use any other module that is proven to perform a good first approximation of the protein structure. This first initializer module can be understood as a fast and low quality solution, which allows one to guide the quantum search to make it simpler. This type of mechanisms are very common classically when the search space is too large and a fast first solution can be obtained.

The search module is an adaptation of the QMS algorithm to the protein folding problem. As in the previous cases, this module runs a QM-H circuit in which the energy values of the proteins are loaded and a search is performed until the minimum energy structure is found.

To calculate the energy of each possible protein structure, using the evaluation function, the PSI4 software tool [120] was used. This tool receives a set of amino acids with a given structure and returns its energy. As in previous use cases, this energy calculation could be done by a quantum algorithm and integrated into the algorithm itself, but the capabilities of quantum devices do not allow it.

However, and unlike other use cases, this quantum algorithm of protein energy calculation is really complex and expensive to implement, being the development of these algorithms a line of research in itself. For this reason, the following part II is devoted to study the complexity of quantum chemistry algorithms with classical algorithms and their application in a use case.

In order to validate QFold as a valid solution, it was executed in real quantum hardware. Due to the limitations, only a reduced size version of the problem was tested but it is useful as a proof-of-concept. This simulation was done using the quantum device IBMQ Casablanca with 7 qubits and quantum volume of 32. Around 20.000 circuits were run just to ensure that the results were valid, avoiding the noise. Besides, it was necessary to use a library, *MITIQ* to mitigate the errores in the quantum hardware.

7.3 Results

- ✓ Application of QMS to a high impact use case where classical computing is very limited.
- ✓ Polynomial quantum advantage over the classical M-H algorithm in the protein folding problem.
- ✓ Quantum algorithm with potential to assist or refine the solutions of the classical algorithm.
- ✓ The first quantum algorithm that uses a realistic model in the protein folding problem, as opposed to quantum algorithms that represent the protein with lattice models.
- ✓ QFold architecture is fully modular for both the classical DL initializer and the quantum search module, allowing the use of different initialization and search algorithms.
- ✓ Execution of M-H quantum algorithm with different beta schemes that allow to study its evolution and behavior on an optimization problem.
- ✓ Test of the QMS algorithm adapted to a use case, called QFold, on a NISQ quantum chip to confirm the advantage results observed in the simulator.

- ✓ Inclusion of QFold into Amazon-AWS repository as a case of study. QFold was consired by AWS experts, as a valuable use case of quantum computing applied to the protein folding problem. It allows any user to use it for their own research.

TFermion: A non-Clifford gate cost assessment library of quantum phase estimation algorithms for quantum chemistry

Pablo A. M. Casares[1], Roberto Campos[1,2], and Miguel A. Martin-Delgado[1,3]

[1]Departamento de Física Teórica, Universidad Complutense de Madrid.
[2]Quasar Science Resources, SL.
[3]CCS-Center for Computational Simulation, Universidad Politécnica de Madrid.

Quantum Phase Estimation is one of the most useful quantum computing algorithms for quantum chemistry and as such, significant effort has been devoted to designing efficient implementations. In this article, we introduce TFermion, a library designed to estimate the T-gate cost of such algorithms, for an arbitrary molecule. As examples of usage, we estimate the T-gate cost of a few simple molecules and compare the same Taylorization algorithms using Gaussian and plane-wave basis.

Contents

Pablo A. M. Casares: pabloamo@ucm.es
Roberto Campos: robecamp@ucm.es
Miguel A. Martin-Delgado: mardel@ucm.es

1 Introduction

Among the different applications found for quantum computing, the original aim of using quantum computers to simulate quantum systems and dynamics [24] still stands out as the most promising one. The reason is twofold: first, a quantum computer can encode the state of the system without needing approximations; and second, since the evolution of (closed) quantum systems is unitary, simulating it is rather natural.

Specifically, quantum computing might be particularly useful to prepare ground states of electronic Hamiltonians and find out their energies. Consequently, they can be employed in a multitude of chemical and material science problems where the ground state energy plays a key role. This includes for instance computing chemical reaction rates [58, 69], and analyzing battery properties [21, 35] or biological enzymes [28].

There exist classical computing techniques able to tackle these problems, most notably Density Functional Theory (DFT) [37]. However, they often rely on approximations, for instance, the Kohn-Sham exchange-correlation parametrized functional, which may struggle to achieve the high accuracy required in some of the problems above. For example, chemical reaction rates depend exponentially on differences in energy. In contrast, the well-known technique Quantum Phase Estimation (QPE) in principle allows achieving the high precision required by these applications. To understand how it works, remember that the Schrodinger equation dictates how a quantum system evolves according to its Hamiltonian,

$$\hat{H}\left|\psi\right\rangle = i\hbar\frac{d}{dt}\left|\psi\right\rangle. \tag{1}$$

If we assume for simplicity that such Hamiltonian is time independent, we can write

$$\begin{aligned}\psi(x,0) &= \sum_n a_n \psi_{E_n}(x) \Rightarrow \\ \psi(x,t) &= \sum_n a_n e^{-iE_n t/\hbar}\psi_{E_n}(x),\end{aligned} \tag{2}$$

for $\psi_{E_n}(x)$ an eigenstate, and E_n the corresponding eigenvalue. We are interested in the ground state energy E_0. Note how the eigenvalues became a phase. To recover it, we can use an inverse Quantum Fourier Transform that will encode such phase in the computational basis, from where it can be readily read out.

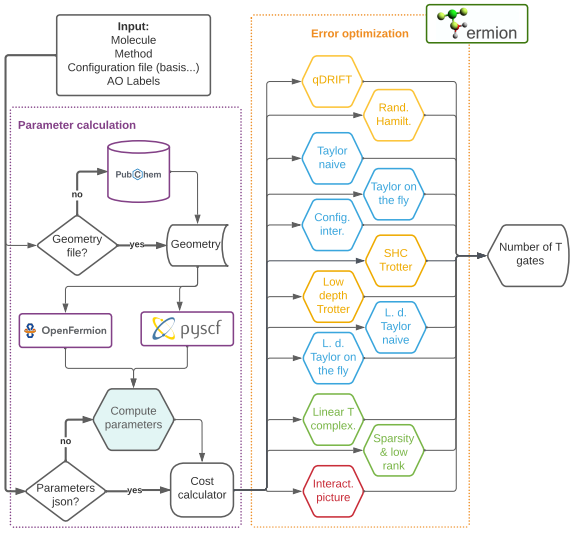

Figure 1: Flowchart of the architecture of our library, divided into two parts: the first one centered on the computation of the parameters needed for the cost estimate; and a second one on using such parameters to compute the number of T-gates. Methods are colored according to the Hamiltonian simulation technique in figure 2.

To implement such an algorithm we need to specify how to implement the quantum Fourier transform, and also the Hamiltonian simulation e^{-iHt}. The former is rather straightforward and can be found in Ref. [53] for example, but the latter is more involved. Furthermore, to obtain a binary description with b bits of the eigenvalue and probability of failure p_f, quantum phase estimation will need to implement $(e^{-iHt})^{2^j}$ for j in the range $1, \ldots, b + \left\lceil \log_2\left(\frac{1}{2} + \frac{1}{2p_f}\right)\right\rceil$ [19]. In other words, it will require the implementation of several time segments that scales with the inverse precision, $O(1/\epsilon_{QPE})$. For this reason, it is important to be able to implement Hamiltonian simulation efficiently.

Such a Hamiltonian might be accessed by the quantum computer in different ways. For electronic Hamiltonians, the most convenient one often is in the form of Linear Combination of Unitaries (LCU). In such framework, we decompose $H = \sum_j a_j H_j$, for a_j some real positive coefficients, and H_j the unitaries, often Pauli string-like operators.

Given such access, there are also various methods to simulate the Hamiltonian evolution. The first way discovered was the Trotter method [2, 42], and soon others such as Taylor series (or Taylorization) [9], Qubitization [43, 45] and Interaction picture simulation (or Dyson series) [34, 44] followed. These Hamiltonian simulation techniques, reviewed in section 3, are the backbone of the quantum phase estimation algorithm. Their objective is to lower as much as possible the computational cost of QPE, so large quantum sys-

tems can be simulated to high precision in reasonable amounts of time, once fault tolerant quantum computers become available. The library we present in this article, TFermion, is an attempt to standardize and automatise the computation of the cost of several quantum phase algorithms in the literature.

However, to use quantum phase estimation, we need to prepare states with a large overlap with the ground state. This will translate into a high probability of measuring the actual ground state energy, and upon success will also project the system into the ground state. Unfortunately, it is known that preparing a representation of the ground state of a 2-body quantum Hamiltonian is Quantum Merlin Arthur (QMA) complete [33], that is, a quantum computer can efficiently verify the solution, but not necessarily efficiently compute it. In other words, finding the ground state of a 2-body Hamiltonian is not in the Bounded Quantum Polynomial-time (BQP) complexity class, the class of problems a quantum computer can solve in polynomial time. Nevertheless, this does not imply either that we cannot propose algorithms to solve it as efficiently as possible [26, 41]. While it is known that the general 2-body ground state preparation is QMA-complete, there is hope that the specificity of electronic Hamiltonians will make it easier to solve at least heuristically. In fact, over the years significant effort has been devoted to the formulation of shallow-depth NISQ ansätze [76] to prepare ground states such as the Imaginary Time Evolution ansatz [47] and the Variational Quantum Eigensolver (VQE) with Unitary Coupled-Cluster [54], adaptative [29], and hardware-efficient [32] ansätze.

Similarly, some effort has been devoted to resource estimates of particular applications [35, 58, 69], but to the best of our knowledge, no software library has been developed to allow a principled comparison between methods. This is a gap that TFermion aims to fill with the following contributions:

First, while newer algorithms often provide a specific non-Clifford gate and qubit count, older ones only give asymptotic complexity estimates (see figure 2). Our article aims to estimate the T-gate cost of older and some of the newer algorithms, with a molecule of choice from the software users. We believe this will be helpful to more quickly carry out research for both academics and industry. Not only that, but our software automatically performs optimization based on the different error sources to minimize the cost, and low-rank approximations [11].

Second, as an example of use of our software, we address the question of whether Gaussian basis functions or plane waves are more convenient to simulate molecules, comparing the same Taylor series algorithms with a different basis. This comparison is not definitive because the error arising from a finite-size basis is difficult to estimate. However, we can give an idea of which algorithms might be more beneficial according to some rough estimates of how many plane waves are required to simulate a system to the same precision than if one were to use Gaussian basis [6]. We furthermore provide researchers with the possibility to carry out a similar comparison but deciding the multiplicative factor for plane waves to represent a similar precision or if the comparison is not the objective, the number of plane waves too.

In TFermion, so far we have focused on T-gates as we believe that non-Clifford gates represent a more significant bottleneck than the number of qubits. Nevertheless, in the future, we expect to add this functionality and additional algorithms to the library. The article itself is structured as follows: first, we give an overview of the library and how it works. Then, in section 3 we briefly explain some of the techniques for Quantum Phase Estimation and Hamiltonian simulation, including figure 2 and table 3 to help the reader understand the development and relation between different algorithms. In section 4, we give examples of how our software might be used, including the second contribution listed above. We then summarize the conclusions and present future work. Finally, in each appendix, we quickly describe one of the techniques studied in this paper, that can be used in combination with the original references to understand the cost estimation functions of TFermion.

2 The TFermion library

The first and main contribution of this article is a software library called TFermion that automatizes the estimation of T-gate cost of running a variety of Quantum Phase Estimation algorithms proposed in the literature during the last years, over arbitrary molecular geometries.

We envision several use cases of our library:

1. It could serve as a quick assessment for the feasibility of concrete QPE experiments once error-corrected quantum computers become available, such as those centered in particular scientific or industrial use cases [35, 58].

2. It can also help make comparisons between systems and methods. In particular, it allows comparing the impact of the chosen Hamiltonian simulation technique, or the chosen basis.

The result provided by our library though must be interpreted as an approximation to the true value, as the final implementation will be heavily optimized, both at a hardware and software level. Our library, in contrast, aims to be more modular and system-agnostic, but we nevertheless provide built-in error optimization. It is well known that different error sources impact the final precision and gate cost in different ways. As such we have aimed at standardizing the way error sources are treated and optimized (see table 1).

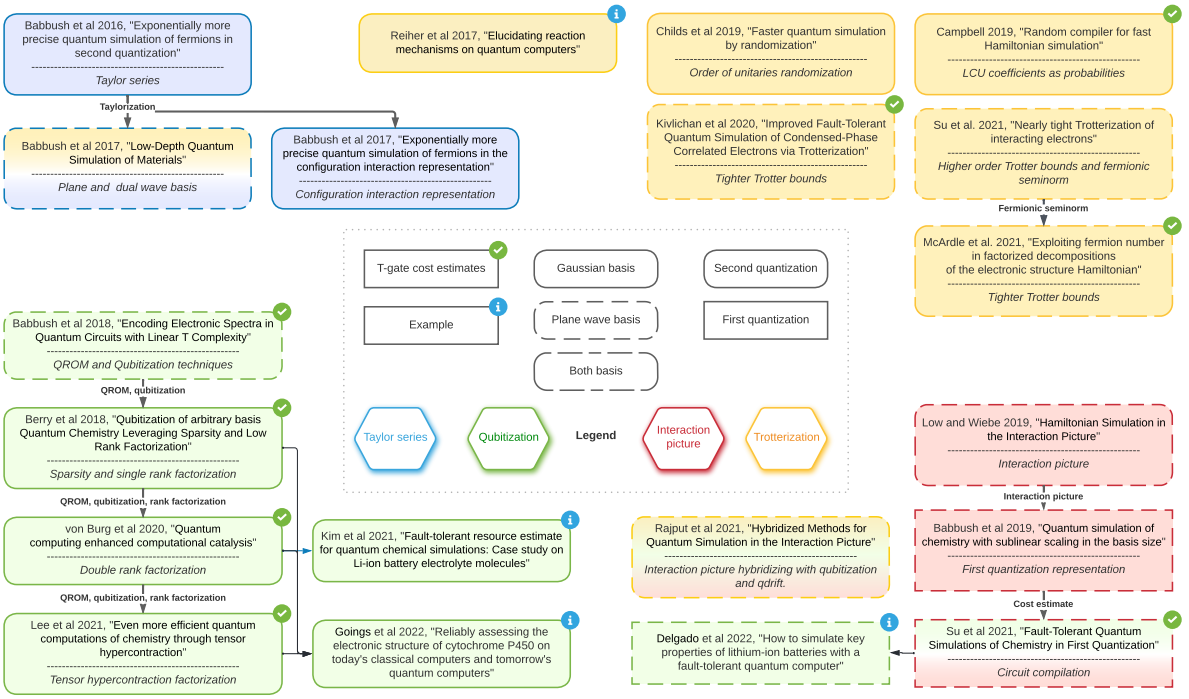

Figure 2: Diagram showing some of the main techniques involved in the development of post-Trotter Quantum Phase Estimation Techniques. Not shown in the picture but of great importance are the articles crystallizing the concept of 'Taylorization' [9] and 'qubitization', [44].

Error	Mathematical definition	Where does it appear?
ϵ_{QPE}	$\epsilon_{QPE} = \lambda 2^{-n}$, n precision bits in the QPE algorithm, and λ the 1-norm of the Hamiltonian.	Due to the Phase Estimation.
ϵ_{HS}	Trotter: $\left\|\left\|e^{-iHt/r} - \mathcal{S}_p(H;t/r)\right\|\right\|_2 \leq W_p \left(\frac{t}{r}\right)^{p+1} \leq \frac{\epsilon_{HS}}{r}$ [48]. Taylor: $\left\|\left\|\Pi_0 A \left\|0\right\rangle \left\|\psi\right\rangle - \left\|0\right\rangle U_r \left\|\psi\right\rangle\right\|\right\|_2 \leq \frac{\epsilon_{HS}}{r}$ [9]. Dyson: $\left\|\left\|W - \mathcal{T}[e^{-i\int_0^{t/r} H(s)ds}]\right\|\right\|_2 \leq \frac{\epsilon_{HS}}{r}$ [44].	In Hamiltonian Simulation via Trotter, Taylor or Dyson series decomposition of $e^{-iH\tau}$.
ϵ_H	$\left\|\int_\Omega f(\mathbf{x})d\mathbf{x} - \sum_{\mathbf{x}\in\Omega} f(\mathbf{x})(\Delta\mathbf{x})^d\right\| < \epsilon_H$, with $d = dim(\Omega)$	Error from the approximation of integrals by Riemannian sums.
ϵ_S & ϵ_{SS}	$\|\|U - R_z(\theta)\|\|_2 \leq \epsilon_{SS}$ [60] (Using operator norm)	In the synthesis of single rotations ϵ_{SS} and their sum, ϵ_S.
ϵ_{tay}	Defined as in Taylor's theorem.	Due to Taylor error series (and others) in arithmetic operations.

Table 1: Notation for the main sources of error that we take into account in the article and software library. Additional minor sources may appear sporadically in single articles. The norm 2 used in all cases above is the operator norm. The other algorithms used to compute arithmetic operations are the Babylon algorithm for the square root, and CORDIC algorithm for the sine or cosine. $\mathcal{S}_p(H;t/r)$ stands for the order p Trotter step, and $W_p = O\left(\max_{\boldsymbol{i}}[[\ldots[H_{\gamma_{i_1}}, H_{\gamma_{i_2}}], H_{\gamma_{i_3}}], \ldots], H_{\gamma_{i_{p+1}}}]\right)$ the commutator terms that bound the final error [63].

While not the main objective of our article, we also believe our work may help provide a more standardized treatment across methods, and as a consequence help better understand the choices in the Hamiltonian simulation, basis, or fermion-qubit mapping used.

One feature of our library is that it currently contains older than 1-year-old methods, and as such some excellent work [39, 62, 69] has not yet been included. There are two reasons: the first and most obvious one is that including new methods represents a significant amount of effort, and we believe these updates can be done later on. The second is that while for the latest methods T-gate estimates are more common, for older ones often only the complexity estimates are available. While this makes sense as the latest methods might be more useful for industrial processes, we believe that understanding well different techniques and not only the bleeding edge ones can be of significant scientific interest.

Additionally, our software was developed following a modular architecture with an easy procedure to include new methods. The process to add a newer method or updating an existing one requires two main steps: first making sure that the molecular parameters required are already calculated by some of the provided methods, or adding new ones in `molecule.py`; then create a new T-gate cost estimation function and call it from the class `Cost calculator`. The philosophy underlying this architecture is to keep TFermion updated timely and give the authors of the new methods the possibility to add their own T gate cost estimation to show practical examples of their work and make it more accessible.

The use of the library is rather straightforward: the user only needs to provide a `molecule name`, a `method` and optionally some atomic orbital labels (`ao labels`) to be used within the active space selection method AVAS [59] to restrict the calculation and make it more efficient. This should be supplemented within a configuration file with the Gaussian basis to be used. If the method requires plane waves to be used, the system will by default approximate the number of basis functions as the thumb rule of 100 times more plane waves than Gaussian waves [6]. Alternatively, the user might provide this and other molecular parameters (eg λ, N...) in a JSON file under the name `[molecule name]_[basis].json`. A flowchart of the working of the library can be seen in figure 1.

As it is shown in figure 1, TFermion is executed through a `main` module which receives the molecule name, the QPE method, and optionally also the ao-labels to select an active space using AVAS. It starts with the `molecule` module creating a `molecule` instance, which is passed together with the method to `cost calculator`. The latter one calls either Gaussian or Plane Waves `molecule` methods to calculate all necessary parameters. Finally, `cost calculator` minimizes the cost depending on the error sources on the selected method, and sends the result back to `main`.

Operation	Cost
Addition & subtr. [27]	$4n$
Multiplication [52]	$21n^2$
Division [67]	$14n^2 + 7n$
Comparison [20]	$8n$
Multi-controlled Not [8]	$16(m-2)$ m controls
Rotation synthesis [60]	$10 + 12\lceil \log_2 \epsilon_{SS}^{-1} \rceil$, $SU(2)$ $10 + 4\lceil \log_2 \epsilon_{SS}^{-1} \rceil$, R_z
State synthesis [61]	$2^{n+1} - 2$ arbitrary rotations

Table 2: Cost of basic arithmetic operators in T gates unless otherwise stated, omitting additive $O(1)$ factors. If the rotation synthesis is controlled, the cost will be multiplied by 2 for R_x, R_y and R_z gates, as given by Lemma 5.4 in [8]. Notice that $HR_zH = R_x$, while R_y and R_z are particular cases of the unitary W in that Lemma. Finally, for general controlled rotations the cost will be thrice the synthesis cost instead of twice.

TFermion manages four types of data:

- **Molecule:** A class created to save all the molecular data, including geometric information obtained [12] used to compute the electronic integrals using Pyscf [64].
- **MolecularData:** An instance from the OpenFermion class [49], necessary to get all parameters from the Hamiltonian and save them into instance `molecule` as attributes.
- **Error values:** Different QPE methods have different error sources, whose sum must not exceed a given threshold. By default we will use the `chemical accuracy` value of 0.0016 Hartrees [17]. TFermion optimizes error values to minimize the T-gate cost output of that method without exceeding it.
- **T gate cost:** Number of T gates needed to execute the selected method, as well as the time required to synthesize the corresponding number of magic states. Calculating this value is the main goal of our library.

Certain calculations in the library are computationally and memory intensive. The reason for this is that as the number of basis functions grows, so does the size of the one and two-body Hamiltonian terms, but does so at least quadratically. This is reflected especially in the plane wave case for molecules, where the larger number of plane waves is due to the need for significantly more basis functions. Nevertheless, an effort has been put into making the calculations relatively efficient, making use of some new techniques [38].

Algorithm	Simulation	Quantization	Basis	Encoding
Random Hamiltonian [15, 18]	Trotter	2nd quantization	Gaussian	Jordan-Wigner
qDRIFT [14, 15]	Trotter-related	2nd quantization	Gaussian	Jordan-Wigner
Taylorization 'database' [3]	Taylor series	2nd quantization	Gaussian	Jordan-Wigner
Taylorization 'on-the-fly' [3]	Taylor series	2nd quantization	Gaussian	Jordan-Wigner
Configuration Interaction [4]	Taylor series	1st quantization	Gaussian	Slater determinant
Low-depth 'Trotter' [6]	Trotter	2nd quantization	Plane waves	Jordan-Wigner
Low-depth 'Taylor database' [6]	Taylor series	2nd quantization	Plane waves	Jordan-Wigner
Low-depth 'Taylor on-the-fly' [6]	Taylor series	2nd quantization	Plane waves	Jordan-Wigner
Interaction picture [45]	Dyson series	2nd quantization	Plane waves	Jordan-Wigner
Sublinear scaling inter. pict. [7, 62]	Dyson series	1st quantization	Plane waves	Slater determinant
Sublinear scaling qubitization [7, 62]	Qubitization	1st quantization	Plane waves	Slater determinant
Linear T complexity [5]	Qubitization	2nd quantization	Plane waves	Jordan-Wigner
Sparsity and low rank [11]	Qubitization	2nd quantization	Gaussian	Jordan-Wigner
Double factorization [69]	Qubitization	2nd quantization	Gaussian	Jordan-Wigner
Tensor hypercontraction [39]	Qubitization	2nd quantization	Gaussian	Jordan-Wigner
Hybridized method [57]	Trotter & Dyson	2nd quantization	Plane waves	Jordan-Wigner

Table 3: Recent Hamiltonian simulation methods, named after the techniques they use, or the title of the corresponding article, explaining them for efficient Hamiltonian simulation and Quantum Phase Estimation. Notice that qDRIFT, Random Hamiltonian and Hybridize method do not specify the basis or the Fermionic encoding, but the ones we indicate seem to be the most obvious: in the case of qDRIFT and Random Hamiltonian because they are the simplest choice, while in the Hybridized method, it inherits the plane wave structure from the Interaction Picture. Recent work on Trotter Hamiltonian simulation [15, 36, 48, 62] has focused on bounding commutator error terms on a different basis, rather than new methods.

Finally, let us briefly mention what our software does not cover yet. It only provides cost estimates for T-gate count, as it is well known that the magic state distillation required to perform the T-gate often carries the largest cost in 2 the dimensional surface code, which nevertheless exhibits a large threshold. Alternatively, there are codes in 3D, like topological color codes [13] that avoid magic state distillation, and may provide new ways to improve this counting, but they require more qubits for similar distance codes. Furthermore, the Clifford gate count may depend on the specific chip connectivity, and for that reason, we have preferred to ignore it here. Finally, while we believe that the qubit count is important, the number of gates required may provide a more significant constraint in the long term due to the time required to perform the algorithms, as these approaches usually require on the order of 10^2 to 10^4 qubits for realistic targets [35, 58, 62].

The cost of ground state preparation, while significant, is left for future work too. Rough estimates may be possible to obtain for moderately sized systems, using low precision QPE to project the system into the ground state [10].

3 Quantum phase estimation techniques

In this section, we give a quick overview of the main techniques used in the literature to perform quantum phase estimation. Quantum phase estimation requires two main ingredients: the use of a controlled Hamiltonian simulation method and sometimes an inverse Quantum Fourier Transform (QFT). While the original Quantum Phase Estimation protocol did use QFT [25, 53], more modern versions such as Bayesian Quantum Phase Estimation avoid it [75]. This latter approach has also the property of being parallelizable, implementable with minimal classical postprocessing, and requires fewer qubits. However, its cost scales as $\frac{4.7\lambda}{\epsilon_{QPE}}$ instead of the theoretical optimum of $\frac{\pi\lambda}{\epsilon_{QPE}}$ [75]. Since the extra cost of the quantum Fourier transform and the qubits it requires are often negligible, we will instead assume we are using the classical version with a slightly lower cost. We will now explain the other main part, the different Hamiltonian simulation techniques.

3.1 Trotter

Let us assume we want to simulate H for a Linear Combination of Unitaries decomposition $H = \sum_\gamma w_\gamma H_\gamma$. The difficulty is that since the different unitaries H_γ do not need to commute, we cannot write $e^{-iHt} = \prod_\gamma e^{-iw_\gamma H_\gamma t}$. Instead, using the product of

Hamiltonian simulation as we have just done introduces an error $O(\sum |[H_{\gamma_1}, H_{\gamma_2}]|t^2)$ that depends on the commutator.

To handle this error, within the scheme of Trotter, there are two strategies. The first one is to divide the evolution in short time segments so we can quadratically suppress the error. In other words, we implement

$$e^{-iHt} = \left(\prod_{\gamma} e^{-iw_\gamma H_\gamma t/r}\right)^r + O\left(\sum |[H_{\gamma_1}, H_{\gamma_2}]|t^2/r\right). \tag{3}$$

Alternatively, one may attempt to find higher order Trotter formulas that further suppress the error. For example, if (3) is the first order formula, then

$$e^{-iHt} = \left(\left(\prod_{\gamma=1}^{\Gamma} e^{-iw_\gamma H_\gamma t/2r}\right)\left(\prod_{\gamma=\Gamma}^{1} e^{-iw_\gamma H_\gamma t/2r}\right)\right)^r + O\left(\sum |[[H_{\gamma_1}, H_{\gamma_2}], H_{\gamma_3}]|t^3/r^2\right) \tag{4}$$

is the second order one. Higher-order formulas are known, but they also become more convoluted to implement. Another possibility is to use classical randomization of the order in which each of $w_\gamma H_\gamma$ appears in the Hamiltonian, in each evolution segment [18], or to apply Hamiltonian simulation of a random H_γ for fixed amounts of time, with probabilities given in by w_γ/λ for $\lambda = \sum w_\gamma$ [15]. The latter method is called 'qDRIFT' and is explored in appendix A together with a second-order randomized Trotter simulation. Other randomized methods have been explored too [70].

There has also been effort devoted to tightly bounding the commutators to reduce the number of segments [16, 36, 48, 63]. Of these, one with a favorable scaling number of basis functions, $O(N^3)$, is the so-called 'SHC bound' for dual wave basis Hamiltonian [48, 63]. It is implemented as the method `shc_trotter` in our library and can be found in appendix F. Finally, Trotter simulation has historically been one of the first methods to be used to estimate resource estimates, including the famous FeMoco study [58], and later ones [22].

3.2 Taylor series

Methods invented after Trotterization are usually called post-Trotter, and their objective is to lower the Hamiltonian simulation error dependence, ϵ_{HS}, from polynomial to polylogarithmic. Taylor series simulation or Taylorization aims to expand the evolution operator of a small time segment as a Taylor series

$$U_r = e^{-iHt/r} \approx \sum_{k=0}^{K} \frac{1}{k!}(-iHt/r)^k = \sum_{k=0}^{K}\sum_{l_1,\ldots,l_k=1}^{L} \frac{(-it/r)^k}{k!} a_{l_1}\ldots a_{l_k} H_{l_1}\ldots H_{l_k}. \tag{5}$$

This expression is a Linear Combination of Unitaries (LCU), $U_{LCU}^{\text{Tay}} = \sum_{l=0}^{L} b_l U_l$. To implement it, one introduces operators

$$\text{Prepare} : |0\rangle \mapsto \sum_l \sqrt{b_l}\,|l\rangle\,, \tag{6}$$

$$\text{Select} : |l\rangle\,|\psi\rangle \mapsto |l\rangle\,U_l\,|\psi\rangle\,, \tag{7}$$

and defines $U_{LCU}^{\text{Tay}} = (\text{Prepare}^\dagger \otimes \mathbf{1})\text{Select}(\text{Prepare} \otimes \mathbf{1})$. Since U_{LCU}^{Tay} has some failure probability in recovering $|0\rangle$ in the first register, it is customary to use (oblivious) amplitude amplification [9], that reduces the error to ϵ_{HS}/r in each segment.

3.3 Block encoding and qubitization

Similarly, the Hamiltonian often takes the form of a linear combination of unitaries $H = \sum a_l H_l$, from which we can create as the block-encoding operator

$$U_{LCU} = \begin{pmatrix} H/\lambda & \cdot \\ \cdot & \cdot \end{pmatrix}, \tag{8}$$

with new Prepare and Select operators

$$\text{Prepare} : |0\rangle \mapsto \sum_l \sqrt{a_l}\,|l\rangle\,, \tag{9}$$

$$\text{Select} : |l\rangle\,|\psi\rangle \mapsto |l\rangle\,H_l\,|\psi\rangle\,. \tag{10}$$

Using them, we obtain,

$$U_{LCU}\,|0\rangle\,|\psi\rangle = |0\rangle\,\frac{H}{\lambda}\,|\psi\rangle + \sqrt{1 - \frac{\|H\,|\psi\rangle\,\|}{\lambda}}\,|(0,\psi_\lambda)^\perp\rangle\,. \tag{11}$$

However, as we saw this LCU implementation has some probability of failure, which requires amplitude amplification to suppress. An alternative is to construct a quantum walk operator Q with the same spectrum. This is done via a procedure called qubitization [45]. In the case where the corresponding $U^2 = 1$, as is the case for $U_{LCU} = \text{Prepare}^\dagger \cdot \text{Select} \cdot \text{Prepare}$, it can simply be implemented as [45, Corollary 9]

$$Q = \underbrace{\text{Prepare}(2\,|0\rangle\,\langle 0| \otimes \mathbf{1} - \mathbf{1})\text{Prepare}^\dagger}_{R} \cdot \text{Select}. \tag{12}$$

Q implements a Grover rotation in each eigenspace

$$\begin{aligned} Q\,|0\rangle\,|\psi_k\rangle &= \cos(\theta_k)\,|0\rangle\,|\psi_k\rangle - \sin(\theta_k)\,|(0,\psi_k)^\perp\rangle\,, \\ Q\,|(0,\psi_k)^\perp\rangle &= \cos(\theta_k)\,|(0,\psi_k)^\perp\rangle + \sin(\theta_k)\,|0\rangle\,|\psi_k\rangle\,, \end{aligned} \tag{13}$$

for $\cos\theta_k = \frac{E_k}{\lambda}$. In other words, Q is a quantum walk operator

$$Q = \bigoplus_k \begin{pmatrix} \frac{E_k}{\lambda} & -\sqrt{1-\frac{E_k^2}{\lambda^2}} \\ \sqrt{1-\frac{E_k^2}{\lambda^2}} & \frac{E_k}{\lambda} \end{pmatrix}_k . \tag{14}$$

Diagonalizing the subspace spanned by $\{|0\rangle|\psi_k\rangle, |(0,\psi_k)^\perp\rangle\}$, we might write $Q_{LCU} = \bigoplus_k \left(e^{i\theta_k}|\theta_k\rangle\langle\theta_k| + e^{-i\theta_k}|-\theta_k\rangle\langle-\theta_k|\right)$. We can use this operator to create a Chebyshev series that approximates e^{-iHt} [44], with a technique called quantum signal processing [43]. However, it is more straightforward to apply phase estimation directly over $\pm\theta_k$ [10]. Then, computing $\cos(\theta_0)$ we recover the ground state energy.

Additionally, qubitization has the advantage that $RQR = Q^\dagger$, so using this trick we can duplicate the implemented phase with almost no extra cost, so the prefactor in the cost falls from $\frac{\pi\lambda}{\epsilon_{QPE}}$ to $\frac{\pi\lambda}{2\epsilon_{QPE}}$ [5]. Qubitization is often used in combination with QROM and factorization techniques [5, 11, 39, 69], but has also been used in first quantization [7, 62].

3.4 Interaction picture and Dyson series

While the qubitization method is optimal concerning the Hamiltonian simulation error, an alternative approach is to find ways to decrease the 1-norm λ of the Hamiltonian H. Let us assume that $H = A + B$ such that $\|A\| \gg \|B\|$. In the interaction picture, $H_I(t) = e^{iAt}B(t)e^{-iAt}$, so in this framework, the norm of the Hamiltonian decreases to $\|B\|$, and therefore the phase estimation may be cheaper to implement. In this picture, the Hamiltonian simulation is implemented as

$$|\psi(t)\rangle = e^{-iAt}\mathcal{T}\left[e^{-i\int_0^t H(s)ds}\right]|\psi(0)\rangle, \tag{15}$$

where $\mathcal{T}$ denotes time ordering. While the e^{-iAt} might be easy to implement if all unitary operators in LCU decomposition of A commute, the time ordered exponential is more difficult to implement. This might be done with a Dyson series

$$\begin{aligned} U(t) &= \mathcal{T}\left[e^{-i\int_0^t H(s)ds}\right] = \sum_{k=0}^{\infty}(-i)^k D_k \\ D_k &= \frac{1}{k!}\int_0^t \ldots \int_0^t \mathcal{T}[H(t_k)\ldots H(t_1)]d^k t, \end{aligned} \tag{16}$$

that similarly to the Taylor series approach, bears a logarithmic complexity on ϵ_{HS}, and requires to implement the simulation for short time segments and use amplitude amplification at each of them. Operator B is implemented as

$$\frac{B}{\|\lambda_B\|} = \langle 0|\text{Prepare}_B^\dagger \cdot \text{Select}_B \cdot \text{Prepare}_B|0\rangle \tag{17}$$

Using this block encoding of operator B, we can express the block encoding of a time segment of $e^{-i(A+B)\tau}$ as [62]

$$\begin{aligned} e^{-i(A+B)\tau} \approx e^{-iA\tau} \lim_{\substack{K\to\infty \\ M\to\infty}} \sum_{k=0}^{K} \frac{(-i\tau)^k}{M^k k!} \sum_{m_1=0}^{M-1} \cdots \sum_{m_k=0}^{M-1} \\ \left(e^{-i\tau(-1/2-m'_k)A/M} B e^{-i\tau(m'_k-m'_{k-1})A/M} B \ldots \right. \\ \left. B e^{-i\tau(m'_2-m'_1)A/M} B e^{-i\tau(m'_1+1/2)A/M}\right) \\ = \left(\langle 0|\,\text{Prepare}_B^\dagger\right)^{\otimes K} \sum_{k=0}^{K} \frac{(-i\lambda_B\tau)^k}{M^k k!} \sum_{m_1,\ldots,m_k=0}^{M-1} \\ \left(e^{-i\tau(M-1/2-m'_k)A/M}\text{Select}_B e^{-i\tau(m'_k-m'_{k-1})A/M} \right. \\ \text{Select}_B \ldots \text{Select}_B e^{-i\tau(m'_2-m'_1)A/M}\text{Select}_B \\ \left. e^{-i\tau(m'_1+1/2)A/M}\right)\left(\text{Prepare}_B|0\rangle\right)^{\otimes K}, \end{aligned} \tag{18}$$

where $m'_1, \ldots, m'_k$ are the sorted integers from $m_1, \ldots, m_k$. This series might therefore be implemented in a similar fashion as those from Taylor series, and will similarly require amplitude amplification. The Dyson series simulation was first introduced in Refs. [34, 45].

4 Results and an use case example: comparison between different basis functions

In this section, we make use of our library to show usage examples. For that purpose, we will perform two tasks: (1) using the FeMoco Hamiltonian provided in the supplementary material of [39], compute the cost of performing Quantum Phase Estimation with several methods included in the library; and (2) perform T-gate estimation for a few simple molecules with a wide range of methods, making a preliminary comparison of the impact of Gaussian or plane-wave basis in the final T gate count, when using Taylorization as a Hamiltonian simulation method.

FeMoco estimates

Over the last years, FeMoco became a standard benchmark for quantum algorithms [58]. Such a benchmark is realistic and useful because it constitutes the metal active center of an enzyme capable of converting atmospheric nitrogen and hydrogen into ammonia, bypassing the energy-intensive industrial Haber-Bosch process. As the first use case example of our library, we first extend the T-gate cost estimation for several methods. Not only this will help us understand the complexity of previous examples, but will also help check the validity of our results for

FeMoco active space	Reiher et. [58]	Li et. [40]
qDRIFT [15]	7.34e+23	3.62e+23
Rand. Hamilt. [15]	1.32e+28	2.94e+28
Taylor naïve [3]	1.15e+22	1.26e+23
Spars. low-rank [11]	2.36e+13	2.17e+13
w/o failure [11]	4.57e+12	4.12e+12
Results in [11]	4.8e+12	3.9e+12

Table 4: Estimation of number of T-gates required to run different Quantum Phase Estimation algorithms with several algorithms. The second half of the table shows that our library gets similar results as [11], where the 'w/o failure' row indicates we obtain without taking into account failure probability.

the low-rank decomposition method, where previous estimates were available [11].

Using the Taylorization approach [3] has intermediate cost between that of Trotterization (qDRIFT and Random Hamiltonian simulation [15]) and more recent rank-decomposition and qubitization techniques [11]. Furthermore, the last row of table 4 can be compared with the published costs of $1.2{\cdot}10^{12}$ and $9.8{\cdot}10^{11}$ Toffoli gates for both active spaces [11, 40, 58]. Since each Toffoli gate is equivalent to 4 T-gates, our estimation is very close to the numbers originally reported. We believe the small difference is due to a combination of factors. In the first place, the error optimization will usually give more weight to ϵ_{QPE} as it is the most costly error source. Additionally, we take into account some factors such as the Uniform subroutines and an amplitude amplification step in the preparation of uniform superpositions on registers

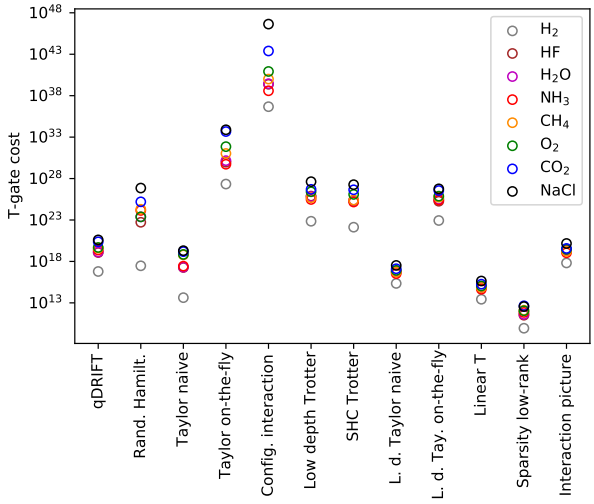

Figure 3: Representation of the results obtained for simple molecules with the results from table 6. We can see that choosing the right method greatly impacts the final cost of the Quantum Phase Estimation algorithm.

N	λ	TFermion	[5] conditions	[5] results
54	5	7.08e+08	2.69e+07	1.80e+07
128	23	4.78e+09	2.26e+08	1.90e+08
250	64	1.96e+10	1.09e+09	1.10e+09
1024	640	5.58e+11	3.88e+10	4.30e+10

Table 5: Replication of the T-gate cost estimates of the `linear_t` method with Jellium, similar to those published in table III from [5]. The third column includes the results with our library, while the fifth those from the original reference [5]. Most of the divergence can be explained because the total error budget has to be allocated between ϵ_{QPE} and ϵ_S, and by considering negligible the rotation synthesis cost. To account for this, the fourth column indicates the results we get if fixed the phase estimation error to $\epsilon_{QPE} = 0.0016$ Hartree, and did not take into account the cost of gate rotation synthesis or failure probability. After this we still do not get the exact results due to other polylogarithmic contributions that the original reference did not considered; but we get quite close.

p and q such that $p \leq q < N/2$ (respectively r and s). We also take a slightly larger number of segments r as described in section 3A of [62], due to the estimation of the phase of $e^{-i\tau \arccos H}$ instead of $e^{-i\tau H}$.

The FeMoco cost of other methods implemented in the library has not been computed, due to the lack of geometry-dependent parameters such as the position of the atoms in FeMoco, or because they were conceived for plane waves instead of gaussian wave functions. In any case, we believe that these results confirm the usefulness of TFermion.

Simple molecules

Next, we run T-gate cost estimates of all the algorithms included in TFermion, with several molecules. As a use-case example, we compare the costs of similar methods on a different basis, something not previously been done in the literature. While these simple molecules can also be analyzed with classical methods, we selected these simple molecules to avoid performing active space selection on them. Of course, selecting such active space in a molecule of scientific interest will represent an important step to making the simulation efficient, but our aim here is to compare the methods rather than obtain novel results for applications of scientific or industrial interest.

The results from our calculations can be seen in table 6. We indicate the median value obtained for each entry after running the procedure 10^3 times. We select the median instead of the average because the results have some inherent stochasticity due to the error sources optimization, but the distribution tends to be skewed to the higher values. We also do not take the lowest value to avoid numerical instability in the ϵ values that may have given rise to unrealistic lower costs.

Method	H_2	HF	H_2O	NH_3	CH_4	O_2	CO_2	NaCl
qDRIFT [15]	6.2e+16	1.2e+19	1.4e+19	2.4e+19	3.9e+19	5.0e+19	2.4e+20	4.0e+20
Rand. Hamilt. [15]	3.0e+17	5.2e+22	2.4e+23	1.4e+24	1.9e+24	2.4e+23	1.6e+25	7.1e+26
Taylor naive [3]	3.0e+13	1.3e+17	1.4e+17	1.9e+17	4.1e+18	4.7e+18	1.1e+19	1.4e+19
Taylor on-the-fly [3]	1.4e+27	5.9e+29	9.4e+29	3.3e+29	6.8e+30	4.6e+31	3.0e+33	4.8e+33
Config. interaction [4]	1.6e+36	2.4e+39	2.8e+39	3.9e+38	1.0e+40	8.3e+40	2.5e+43	4.3e+46
Low depth Trotter [6]	1.2e+23	1.3e+26	1.1e+26	5.0e+25	8.5e+25	4.4e+26	8.4e+26	6.9e+27
SHC Trotter [6, 48]	2.3e+22	3.6e+25	4.2e+25	2.5e+25	4.2e+25	2.0e+26	7.5e+26	3.2e+27
L. d. Taylor naive [6]	3.1e+15	7.8e+16	8.4e+16	4.9e+16	7.6e+16	1.2e+17	1.8e+17	4.7e+17
L. d. Tay. on-the-fly [6]	1.3e+23	2.7e+25	4.7e+25	3.7e+25	8.4e+25	1.1e+26	5.2e+26	8.5e+26
Linear T [5]	3.9e+13	1.0e+15	1.1e+15	6.3e+14	9.7e+14	1.6e+15	2.6e+15	6.3e+15
Sparsity low-rank [11]	**1.2e10**	**4.6e11**	**6.0e11**	**1.0e12**	**1.8e12**	**1.5e12**	**6.3e12**	**5.3e12**
Interaction picture [45]	1.4e+18	5.7e+19	5.0e+19	2.4e+19	3.6e+19	6.6e+19	8.0e+19	3.3e+20

Table 6: T-gate cost estimates for different molecules and methods obtained using our TFermion, see Fig. 3. The Rank decomposition technique is the most efficient between the analysed methods, closely followed by the plane wave methods using QROM and qubitization ('Linear T') or Taylorization ('Low depth Taylor naïve').

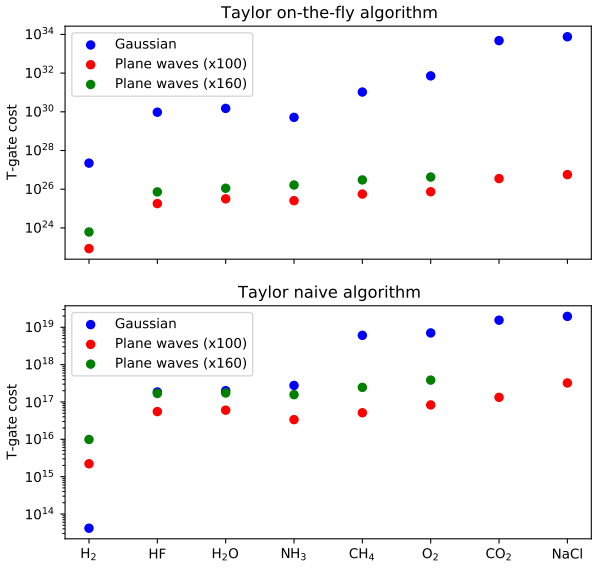

Figure 4: T-gate cost of performing the same algorithms making use of Taylorization as the main Hamiltonian simulation technique, over different molecules. The number of plane waves was chosen to be ≈ 100 or 160 times larger than Gaussian functions as recommended by Appendix E in [6]. The cost of computing the electronic integrals on-the-fly is larger than classically precomputing and loading them. The comparison between Gaussian and plane-wave basis should be taken with care as the error due to finite basis size was not rigorously computed and controlled.

Let us first comment on the results of some methods. The first thing that calls our attention is the large cost of the Configuration Interaction method [4]. We believe this is due to a combination of three factors: the first and most important one is that the condition on the number of segments r imposed by the Lemmas 1-3 in [4] is a very large value, which may be understood as an upper bound rather than a real cost estimate. Secondly, our method to perform the procedure from section 4.1 was not optimized. And thirdly, it also contains a large number of arithmetic operations, similar to those in 'Taylor on-the-fly'. Overall this indicates that the estimates for this method should be treated as an upper bound.

We can also observe that when using a Gaussian basis, Taylor methods are almost always more efficient than Trotter ones and that the cost of using the on-the-fly versions of Taylor is often larger than the naïve one due to the arithmetic operations. The interaction picture algorithm [6] displays a 'similar' complexity as the Taylorization algorithms [3], as both operate on a Gaussian basis and decompose the evolution operator in a Taylor or Dyson series.

The most efficient algorithms among the analyzed ones are those making use of the QROM techniques, [5, 11]. Surprisingly though, the Low depth Taylor naïve [6] achieves the third-best complexity just after the rank-decomposition algorithm [11], and the original article introducing the QROM [5]. We believe the reason for that is that the original article left unspecified the techniques that should be used to implement Prepare and Select, so we have assumed the use of modern QROM techniques [5].

To make this comparison fair, we have, as a rule of thumb, used approximately 100 times as many plane waves as Gaussian wave functions, as it has been suggested for isolated molecules [6]. The Gaussian basis used is the standard 6-31G [31], but this may be changed by the user at will in the configuration file, as well as the multiplicative factor. Using the previously mentioned ratio, we can as an example of usage of our library, compute the cost of the same Taylorization methods with Gaussian and plane waves. The results are shown in figure 4, although these results must be taken with care as we have not controlled the

error introduced by different finite basis sets.

5 Conclusions and future work

Over the last years significant effort has been devoted to creating efficient algorithms for Quantum Phase Estimation and Hamiltonian simulation since the estimation of ground state energy is such a central problem for quantum chemistry and a very natural application of quantum computing. TFermion fills a gap in standardizing and easing the use of such algorithms. It should help academics have a better understanding of algorithms for which no complexity estimates were previously available. The usefulness for the industry is clear too, as it reduces the effort required to quickly iterate over specific use-cases. As examples of usage, we have run calculations with FeMoco and a range of molecules. Among the most interesting results is the fact that using QROM techniques in the plane wave naïve Taylorization method [6] makes it particularly efficient, and we have seen hints that using plane-wave could be more efficient than Gaussian for the same Taylorization techniques in isolated molecules.

However, the effort is far from complete. On one hand, exciting avenues of research remain open, particularly in the use of plane waves [62]. On the other, we aim to improve this library in several dimensions: (1) newer algorithms should be added; (2) our algorithms are designed for molecules instead of materials, where plane-wave methods should become very efficient; (3) TFermion only provides estimates for T-gates so the addition of other metrics such as the number of qubits would be a welcomed addition; and (4) the topic of ground state preparation is barely touched upon but should be considered a prerequisite to estimate the ground state energy.

We believe this is a particularly exciting time to explore how quantum computing can be applied to chemistry and material science. For this reason, we humbly hope that TFermion will become a useful tool to advance the field and find beneficial applications for society.

Code availability

The code for this article can be found at `https://github.com/PabloAMC/TFermion`.

Acknowledgements

We want to thank the very kind explanations of Emiel Koridon of some calculations in one of his articles and beyond. Similarly, we thank answers from Nicolas Rubin and Ryan Babbush on the use of OpenFermion, Joonho Lee on the code from [39], and Antonio Hidalgo, María Jesús Morán, Nelaine Mora and Javier García on quantum chemistry. We acknowledge financial support from the Spanish MINECO grants MINECO/FEDER Projects FIS 2017-91460-EXP, PGC2018-099169-B-I00 FIS-2018, from CAM/FEDER Project No. S2018/TCS-4342 (QUITEMAD-CM), and from Spanish MCIN with funding from European Union NextGenerationEU (PRTR-C17.I1) and Ministry of Economic Affairs Quantum ENIA project. The research of M.A.M.-D. has been partially supported by the U.S. Army Research Office through Grant No. W911NF-14-1-0103. P. A. M. C. thanks the support of a MECD grant FPU17/03620, and R.C. the support of a CAM grant IND2019/TIC17146.

References

[1] Daniel S Abrams and Seth Lloyd. Simulation of many-body fermi systems on a universal quantum computer. *Physical Review Letters*, 79(13):2586, 1997. DOI: https://doi.org/10.1103/PhysRevLett.79.2586.

[2] Alán Aspuru-Guzik, Anthony D Dutoi, Peter J Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707, 2005. DOI: https://doi.org/10.1126/science.1113479.

[3] Ryan Babbush, Dominic W Berry, Ian D Kivlichan, Annie Y Wei, Peter J Love, and Alán Aspuru-Guzik. Exponentially more precise quantum simulation of fermions in second quantization. *New Journal of Physics*, 18(3):033032, 2016. DOI: https://doi.org/10.1088/1367-2630/18/3/033032.

[4] Ryan Babbush, Dominic W Berry, Yuval R Sanders, Ian D Kivlichan, Artur Scherer, Annie Y Wei, Peter J Love, and Alán Aspuru-Guzik. Exponentially more precise quantum simulation of fermions in the configuration interaction representation. *Quantum Science and Technology*, 3(1):015006, 2017. DOI: https://doi.org/10.1088/2058-9565/aa9463.

[5] Ryan Babbush, Craig Gidney, Dominic W Berry, Nathan Wiebe, Jarrod R McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. Encoding electronic spectra in quantum circuits with linear t complexity. *Physical Review X*, 8(4):041015, 2018. DOI: https://doi.org/10.1103/physrevx.8.041015.

[6] Ryan Babbush, Nathan Wiebe, Jarrod R McClean, James McClain, Hartmut Neven, and Garnet Kin-Lic Chan. Low-depth quantum simulation of materials. *Physical Review X*, 8(1):011044, 2018. DOI: https://doi.org/10.1103/physrevx.8.011044.

[7] Ryan Babbush, Dominic W Berry, Jarrod R McClean, and Hartmut Neven. Quantum simulation of chemistry with sublinear scaling in basis size.

npj Quantum Information, 5(1):1–7, 2019. DOI: https://doi.org/10.1038/s41534-019-0199-y.

[8] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457, 1995. DOI: https://doi.org/10.1103/PhysRevA.52.3457.

[9] Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. Simulating hamiltonian dynamics with a truncated taylor series. *Physical Review Letters*, 114(9):090502, 2015. DOI: https://doi.org/10.1103/physrevlett.114.090502.

[10] Dominic W Berry, Mária Kieferová, Artur Scherer, Yuval R Sanders, Guang Hao Low, Nathan Wiebe, Craig Gidney, and Ryan Babbush. Improved techniques for preparing eigenstates of fermionic hamiltonians. *npj Quantum Information*, 4(1):1–7, 2018. DOI: https://doi.org/10.1038/s41534-018-0071-5.

[11] Dominic W Berry, Craig Gidney, Mario Motta, Jarrod R McClean, and Ryan Babbush. Qubitization of arbitrary basis quantum chemistry leveraging sparsity and low rank factorization. *Quantum*, 3:208, 2019. DOI: https://doi.org/10.22331/q-2019-12-02-208.

[12] Evan E Bolton, Yanli Wang, Paul A Thiessen, and Stephen H Bryant. Pubchem: integrated platform of small molecules and biological activities. In *Annual Reports in Computational Chemistry*, volume 4, pages 217–241. Elsevier, 2008. DOI: https://doi.org/10.1016/s1574-1400(08)00012-1.

[13] Hector Bombin and Miguel Angel Martin-Delgado. Topological computation without braiding. *Physical Review Letters*, 98(16):160502, 2007. DOI: https://doi.org/10.1103/physrevlett.98.160502.

[14] Earl Campbell. Shorter gate sequences for quantum computing by mixing unitaries. *Physical Review A*, 95(4):042306, 2017. DOI: https://doi.org/10.1103/physreva.95.042306.

[15] Earl Campbell. Random compiler for fast hamiltonian simulation. *Physical Review Letters*, 123(7):070503, 2019. DOI: https://doi.org/10.1103/PhysRevLett.123.070503.

[16] Earl Campbell. Early fault-tolerant simulations of the hubbard model. *Quantum Science and Technology*, 7(1):015007, 2021. DOI: https://doi.org/10.1088/2058-9565/ac3110.

[17] Yudong Cao, Jonathan Romero, Jonathan P Olson, Matthias Degroote, Peter D Johnson, Mária Kieferová, Ian D Kivlichan, Tim Menke, Borja Peropadre, Nicolas PD Sawaya, et al. Quantum chemistry in the age of quantum computing. *Chemical Reviews*, 119(19):10856–10915, 2019. DOI: https://doi.org/10.1021/acs.chemrev.8b00803.

[18] Andrew M Childs, Aaron Ostrander, and Yuan Su. Faster quantum simulation by randomization. *Quantum*, 3:182, 2019. DOI: https://doi.org/10.22331/q-2019-09-02-182.

[19] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998. DOI: https://doi.org/10.1098/rspa.1998.0164.

[20] Steven A Cuccaro, Thomas G Draper, Samuel A Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv preprint quant-ph/0410184*, 2004. DOI: https://doi.org/10.48550/arXiv.quant-ph/0410184.

[21] Alain Delgado, Pablo Antonio Moreno Casares, Roberto dos Reis, Modjtaba Shokrian Zini, Roberto Campos, Norge Cruz-Hernández, Arne-Christian Voigt, Angus Lowe, Soran Jahangiri, Miguel Angel Martin-Delgado, Jonathan E. Mueller, and Juan Miguel Arrazola. How to simulate key properties of lithium-ion batteries with a fault-tolerant quantum computer. *arXiv preprint arXiv:2204.11890*, 2022. DOI: 10.48550/ARXIV.2204.11890. URL `https://arxiv.org/abs/2204.11890`.

[22] Vincent E Elfving, Benno W Broer, Mark Webber, Jacob Gavartin, Mathew D Halls, K Patrick Lorton, and A Bochevarov. How will quantum computers provide an industrially relevant computational advantage in quantum chemistry? *arXiv preprint arXiv:2009.12472*, 2020. DOI: https://doi.org/10.48550/arXiv.2009.12472.

[23] Andrew J Ferris. Fourier transform for fermionic systems and the spectral tensor network. *Physical Review Letters*, 113(1):010401, 2014. DOI: https://doi.org/10.1103/physrevlett.113.010401.

[24] Richard P Feynman. Simulating physics with computers. In *Feynman and computation*, pages 133–153. CRC Press, 2018. DOI: https://doi.org/10.1201/9780429500459-11.

[25] Alberto Galindo and Miguel Angel Martin-Delgado. Information and computation: Classical and quantum aspects. *Reviews of Modern Physics*, 74(2):347, 2002. DOI: https://doi.org/10.1103/revmodphys.74.347.

[26] Yimin Ge, Jordi Tura, and J Ignacio Cirac. Faster ground state preparation and high-precision ground energy estimation with fewer qubits. *Journal of Mathematical Physics*, 60(2):022202, 2019. DOI: https://doi.org/10.1063/1.5027484.

[27] Craig Gidney. Halving the cost of quantum addition. *Quantum*, 2:74, 2018. DOI: https://doi.org/10.22331/q-2018-06-18-74.

[28] Joshua J Goings, Alec White, Joonho Lee, Christofer S Tautermann, Matthias Degroote, Craig Gidney, Toru Shiozaki, Ryan Babbush, and Nicholas C Rubin. Reliably assessing the electronic structure of cytochrome p450 on today's classical computers and tomorrow's quantum computers. *arXiv preprint arXiv:2202.01244*, 2022. DOI: https://doi.org/10.48550/arXiv.2202.01244.

[29] Harper R. Grimsley, S. Economou, Edwin Barnes, and Nicholas J. Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications*, 10, 2019. DOI: https://doi.org/10.1038/s41467-019-10988-2.

[30] Matthew B. Hastings, Dave Wecker, Bela Bauer, and Matthias Troyer. Improving quantum algorithms for quantum chemistry. *Quantum Information and Computation*, 15(1–2): 1–21, jan 2015. ISSN 1533-7146. DOI: https://doi.org/10.26421/qic15.1-2-1.

[31] Frank Jensen. Atomic orbital basis sets. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 3(3):273–295, 2013. DOI: https://doi.org/10.1002/wcms.1123.

[32] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. Chow, and J. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549:242–246, 2017. DOI: https://doi.org/10.1038/nature23879.

[33] Julia Kempe, Alexei Kitaev, and Oded Regev. The complexity of the local hamiltonian problem. *SIAM Journal on Computing*, 35(5):1070–1097, 2006. DOI: https://doi.org/10.1137/s0097539704445226.

[34] Mária Kieferová, Artur Scherer, and Dominic W Berry. Simulating the dynamics of time-dependent hamiltonians with a truncated dyson series. *Physical Review A*, 99(4):042314, 2019. DOI: https://doi.org/10.1103/physreva.99.042314.

[35] Isaac H Kim, Ye-Hua Liu, Sam Pallister, William Pol, Sam Roberts, and Eunseok Lee. Fault-tolerant resource estimate for quantum chemical simulations: Case study on li-ion battery electrolyte molecules. *Physical Review Research*, 4(2):023019, 2022. DOI: https://doi.org/10.1103/physrevresearch.4.023019.

[36] Ian D Kivlichan, Craig Gidney, Dominic W Berry, Nathan Wiebe, Jarrod R McClean, Wei Sun, Zhang Jiang, Nicholas C Rubin, Austin Fowler, Alán Aspuru-Guzik, et al. Improved fault-tolerant quantum simulation of condensed-phase correlated electrons via trotterization. *Quantum*, 4:296, 2020. DOI: https://doi.org/10.22331/q-2020-07-16-296.

[37] Jorge Kohanoff. *Electronic structure calculations for solids and molecules: theory and computational methods*. Cambridge university press, 2006. DOI: https://doi.org/10.1017/CBO9780511755613.

[38] Emiel Koridon, Saad Yalouz, Bruno Senjean, Francesco Buda, Thomas E O'Brien, and Lucas Visscher. Orbital transformations to reduce the 1-norm of the electronic structure hamiltonian for quantum computing applications. *Physical Review Research*, 3(3):033127, 2021. DOI: https://doi.org/10.1103/physrevresearch.3.033127.

[39] Joonho Lee, Dominic W Berry, Craig Gidney, William J Huggins, Jarrod R McClean, Nathan Wiebe, and Ryan Babbush. Even more efficient quantum computations of chemistry through tensor hypercontraction. *PRX Quantum*, 2(3):030305, 2021. DOI: https://doi.org/10.1103/prxquantum.2.030305.

[40] Zhendong Li, Junhao Li, Nikesh S Dattani, CJ Umrigar, and Garnet Kin-Lic Chan. The electronic complexity of the ground-state of the femo cofactor of nitrogenase as relevant to quantum simulations. *The Journal of Chemical Physics*, 150(2):024302, 2019. DOI: https://doi.org/10.1063/1.5063376.

[41] Lin Lin and Yu Tong. Near-optimal ground state preparation. *Quantum*, 4:372, 2020. DOI: https://doi.org/10.22331/q-2020-12-14-372.

[42] Seth Lloyd. Universal quantum simulators. *Science*, pages 1073–1078, 1996. DOI: https://doi.org/10.1126/science.273.5278.1073.

[43] Guang Hao Low and Isaac L Chuang. Optimal hamiltonian simulation by quantum signal processing. *Physical Review Letters*, 118(1):010501, 2017. DOI: https://doi.org/10.1103/physrevlett.118.010501.

[44] Guang Hao Low and Isaac L Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, 2019. DOI: https://doi.org/10.22331/q-2019-07-12-163.

[45] Guang Hao Low and Nathan Wiebe. Hamiltonian simulation in the interaction picture. *arXiv preprint arXiv:1805.00675*, 2018. DOI: https://doi.org/10.48550/arXiv.1805.00675.

[46] Guang Hao Low, Vadym Kliuchnikov, and Luke Schaeffer. Trading t-gates for dirty qubits in state preparation and unitary synthesis. *arXiv preprint arXiv:1812.00954*, 2018. DOI: https://doi.org/10.48550/arXiv.1812.00954.

[47] Sam McArdle, Tyson Jones, Suguru Endo, Y. Li, S. Benjamin, and Xiao Yuan. Variational ansatz-based quantum simulation of imaginary time evolution. *npj Quantum Information*, 5:1–6, 2018. DOI: https://doi.org/10.1038/s41534-019-0187-2.

[48] Sam McArdle, Earl Campbell, and Yuan Su. Exploiting fermion number in factorized decompositions of the electronic structure hamiltonian.

Physical Review A, 105(1):012403, 2022. DOI: https://doi.org/10.1103/physreva.105.012403.

[49] Jarrod R McClean, Nicholas C Rubin, Kevin J Sung, Ian D Kivlichan, Xavier Bonet-Monroig, Yudong Cao, Chengyu Dai, E Schuyler Fried, Craig Gidney, Brendan Gimby, et al. Openfermion: the electronic structure package for quantum computers. *Quantum Science and Technology*, 5(3):034014, 2020. DOI: https://doi.org/10.1088/2058-9565/ab8ebc.

[50] Mario Motta, Erika Ye, Jarrod R McClean, Zhendong Li, Austin J Minnich, Ryan Babbush, and Garnet Kin Chan. Low rank representations for quantum simulation of electronic structure. *npj Quantum Information*, 7(1):1–7, 2021. DOI: https://doi.org/10.1038/s41534-021-00416-z.

[51] Felix Motzoi, Michael P Kaicher, and Frank K Wilhelm. Linear and logarithmic time compositions of quantum many-body operators. *Physical Review Letters*, 119(16):160503, 2017. DOI: https://doi.org/10.1103/physrevlett.119.160503.

[52] Edgard Muñoz-Coreas and Himanshu Thapliyal. T-count optimized design of quantum integer multiplication. *arXiv preprint arXiv:1706.05113*, 2017. DOI: https://doi.org/10.48550/arXiv.1706.05113.

[53] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[54] Alberto Peruzzo, Jarrod R McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213, 2014. DOI: https://doi.org/10.1038/ncomms5213.

[55] David Poulin, Matthew B Hastings, Dave Wecker, Nathan Wiebe, Andrew C Doherty, and Matthias Troyer. The trotter step size required for accurate quantum simulation of quantum chemistry. *arXiv preprint arXiv:1406.4920*, 2014. DOI: https://doi.org/10.26421/qic15.5-6-1.

[56] David Poulin, Alexei Kitaev, Damian S Steiger, Matthew B Hastings, and Matthias Troyer. Quantum algorithm for spectral measurement with a lower gate count. *Physical Review Letters*, 121(1):010501, 2018. DOI: https://doi.org/10.1103/physrevlett.121.010501.

[57] Abhishek Rajput, Alessandro Roggero, and Nathan Wiebe. Hybridized methods for quantum simulation in the interaction picture. *arXiv preprint arXiv:2109.03308*, 2021. DOI: https://doi.org/10.48550/arXiv.2109.03308.

[58] Markus Reiher, Nathan Wiebe, Krysta M Svore, Dave Wecker, and Matthias Troyer. Elucidating reaction mechanisms on quantum computers. *Proceedings of the National Academy of Sciences*, 114(29):7555–7560, 2017. DOI: https://doi.org/10.1073/pnas.1619152114.

[59] Elvira R Sayfutyarova, Qiming Sun, Garnet Kin-Lic Chan, and Gerald Knizia. Automated construction of molecular active spaces from atomic valence orbitals. *Journal of Chemical Theory and Computation*, 13(9):4063–4078, 2017. DOI: https://doi.org/10.1021/acs.jctc.7b00128.s001.

[60] Peter Selinger. Efficient clifford+t approximation of single-qubit operators. *Quantum Info. Comput.*, 15(1–2):159–180, jan 2015. ISSN 1533-7146. DOI: https://doi.org/10.26421/qic15.1-2-10.

[61] Vivek V Shende, Stephen S Bullock, and Igor L Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006. DOI: https://doi.org/10.1109/tcad.2005.855930.

[62] Yuan Su, Dominic W Berry, Nathan Wiebe, Nicholas C Rubin, and Ryan Babbush. Fault-tolerant quantum simulations of chemistry in first quantization. *PRX Quantum*, 2(4):040332, 2021. DOI: https://doi.org/10.1103/prxquantum.2.040332.

[63] Yuan Su, Hsin-Yuan Huang, and Earl T Campbell. Nearly tight trotterization of interacting electrons. *Quantum*, 5:495, 2021. DOI: https://doi.org/10.22331/q-2021-07-05-495.

[64] Qiming Sun, Timothy C Berkelbach, Nick S Blunt, George H Booth, Sheng Guo, Zhendong Li, Junzi Liu, James D McClain, Elvira R Sayfutyarova, Sandeep Sharma, et al. Pyscf: the python-based simulations of chemistry framework. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 8(1):e1340, 2018. DOI: https://doi.org/10.1002/wcms.1340.

[65] Masuo Suzuki. Fractal decomposition of exponential operators with applications to many-body theories and monte carlo simulations. *Physics Letters A*, 146(6):319–323, 1990. DOI: https://doi.org/10.1016/0375-9601(90)90962-n.

[66] Masuo Suzuki. General theory of fractal path integrals with applications to many-body theories and statistical physics. *Journal of Mathematical Physics*, 32(2):400–407, 1991. DOI: https://doi.org/10.1063/1.529425.

[67] Himanshu Thapliyal, TSS Varun, Edgard Munoz-Coreas, Keith A Britt, and Travis S Humble. Quantum circuit designs of integer division optimizing t-count and t-depth. In *2017 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*, pages 123–128. IEEE, 2017. DOI: https://doi.org/10.1109/inis.2017.34.

[68] Jack E Volder. The cordic trigonometric computing technique. *IRE Transactions on electronic computers*, (3):330–334, 1959. DOI: https://doi.org/10.1109/tec.1959.5222693.

[69] Vera von Burg, Guang Hao Low, Thomas Häner, Damian S Steiger, Markus Reiher, Martin Roetteler, and Matthias Troyer. Quantum computing enhanced computational catalysis. *Physical Review Research*, 3(3):033055, 2021. DOI: https://doi.org/10.1103/physrevresearch.3.033055.

[70] Kianna Wan, Mario Berta, and Earl Campbell. A randomized quantum algorithm for statistical phase estimation. *arXiv preprint arXiv:2110.12071*, 2021. DOI: https://doi.org/10.48550/arXiv.2110.12071.

[71] Dave Wecker, Matthew B Hastings, Nathan Wiebe, Bryan K Clark, Chetan Nayak, and Matthias Troyer. Solving strongly correlated electron models on a quantum computer. *Physical Review A*, 92(6):062318, 2015. DOI: https://doi.org/10.1103/physreva.92.062318.

[72] Steven R White. Hybrid grid/basis set discretizations of the schrödinger equation. *The Journal of Chemical Physics*, 147(24):244102, 2017. DOI: https://doi.org/10.1063/1.5007066.

[73] Steven R White and E Miles Stoudenmire. Multisliced gausslet basis sets for electronic structure. *Physical Review B*, 99(8):081110, 2019.

[74] James D Whitfield, Jacob Biamonte, and Alán Aspuru-Guzik. Simulation of electronic structure hamiltonians using quantum computers. *Molecular Physics*, 109(5):735–750, 2011. DOI: https://doi.org/10.1080/00268976.2011.552441.

[75] Nathan Wiebe and Chris Granade. Efficient bayesian phase estimation. *Physical Review Letters*, 117(1):010503, 2016. DOI: https://doi.org/10.1103/physrevlett.117.010503.

[76] Ruizhe Zhang, Guoming Wang, and Peter Johnson. Computing Ground State Properties with Early Fault-Tolerant Quantum Computers. *Quantum*, 6:761, July 2022. ISSN 2521-327X. DOI: 10.22331/q-2022-07-11-761. URL `https://doi.org/10.22331/q-2022-07-11-761`.

A qDRIFT, a random Hamiltonian trotterization approach

Using Hamiltonian simulation to estimate the energy of chemical configurations can be accomplished through different methods. We will present the main ones that can be chosen from in our software package in the following appendices. We first consider the *Trotter-Suzuki decomposition* [1, 65, 66], where the time evolution of a Hamiltonian $H = \sum_{\gamma=1}^{\Gamma} w_\gamma H_\gamma$, with H_γ being a normalized Hermitian operator and w_γ a non-negative Hamiltonian coefficient, is approximated by

$$e^{-iHt} = e^{-it\sum_\gamma w_\gamma H_\gamma} \approx \left(\prod_{\gamma=1}^{\Gamma} e^{-iw_\gamma H_\gamma t/r}\right)^r. \quad (19)$$

In the limit of $r \to \infty$ the equality is exact. Notice that H and H_γ do not need to be unitary in general, only Hermitian. In contrast, e^{-iHt} is unitary, and since the electronic Hamiltonian can be written in second quantization as a Linear Combination of Unitaries, for the estimation of the cost of this method we will in fact take H_γ to be unitary, as in the rest of the described methods. In this section, we present the qDRIFT and Random Hamiltononian methods, some of the best method that uses the Trotter-Suzuki decomposition [15]. The main idea here is to reduce the complexity of the Trotter Suzuki decomposition above by randomizing the order in which the terms $e^{-iH_\gamma t/r}$ are applied. They suggest to simulate a single unitary $e^{-i\tau H_\gamma}$ randomly from an identical distribution, where $\tau = t\lambda/r$ is fixed, $\lambda = \sum_{\gamma=1}^{\Gamma} w_\gamma$, and the probability of choosing an individual unitary is weighted by the Hamiltonian coefficient w_γ. We further define $\Lambda = \max_\gamma w_\gamma$. This markovian method is referred to as the qDRIFT approach.

The qDRIFT algorithm achieves $O(\lambda^2 t^2/\epsilon_{HS})$ gate complexity, where ϵ_{HS} is the desired precision. This scaling stems from making the zeroth and first-order expansion terms of the qDRIFT quantum channel coincide with the channel that describes the unitary evolution. In contrast, the $2k$-th order (deterministic) Trotter methods have complexity $O(\Gamma^{2+1/2k}(\Lambda t)^{1+1/2k}/\epsilon_{HS}^{1/2k})$ [15]. As a consequence, the qDRIFT algorithm proves advantageous whenever $\lambda \ll \Lambda\Gamma$, which is the case for most electronic structure Hamiltonians, as the majority of terms H_γ possess small coefficients w_γ [11]. On the other hand, qDRIFT will most likely perform worse than higher-order Trotter expansion for large evolution times.

In the following, we will present the number of T gates required for performing the unitary evolution of Eq. (19) through the qDRIFT method and a second order Trotterization method, respectively. The details of this analysis are based on the supplementary material of [15] and consider the problem of estimating the ground state energy E_0 of a Hamiltonian H using quantum phase estimation. The total number of gates n of the form $e^{-i\tau H_\gamma}$ required to estimate the energy of the ground state to an additive error δ_E using qDRIFT is given by [15]

$$n = \frac{\pi^2\lambda^2}{\epsilon_{tot}\delta_E^2}\left(\frac{1+p_f}{p_f}\right)^2, \quad (20)$$

where p_f is the failure probability inherent to the quantum phase estimation algorithm and ϵ_{tot} is the total Trotter error. Similarly, using a second-order random Trotterization, this number scales as [15]

$$n = 8\Gamma^2\frac{1}{\epsilon_{tot}}\left(\frac{\pi\Lambda}{2\delta_E}\right)^{3/2}\left(\frac{1+p_f}{p_f}\right)^{3/2}. \quad (21)$$

To arrive at the cost in terms of T-gates, we need to assess the T-gate cost of simulating a gate $e^{-i\tau H_\gamma}$ and then multiply it by n as given by Eq. (20) and Eq. (21) to give an estimate for the cost of performing qDRIFT and a second-order Trotterization approach, respectively.

The difficulty here is that H_γ will be a string of Pauli operators, so we cannot just implement the rotation in each qubit separately as it is an entangling rotation. Fortunately, we can perform each $e^{-iH_\gamma\tau}$ using Clifford gates and a single C-R_z rotation [30, 51]. This, in turn can be decomposed in two R_z gates using Lemma 5.4 from [8], and each rotation implemented with $\approx 10 + 4\log(\epsilon_{SS}^{-1})$ T-gates [60].

Finally, notice that in the notation of our article, we are taking $\delta_E = 2\epsilon_{QPE}$ and $\epsilon_{tot} = \epsilon_{HS}$. Similarly ϵ_{SS} can be determined by dividing ϵ_S (which is not taken into account in [15]), by the number of unitary Pauli rotations used, $2n$.

B Taylorization-based Hamiltonian simulation

If in the previous appendix we explored the Trotter and Trotter-like methods for Hamiltonian simulation, from now on we would like to focus on so-called post-Trotter methods that allow avoiding having polynomial complexity in the Hamiltonian simulation precision ϵ_{HS}^{-1}. We will start with a method called Taylorization [3].

B.1 Method explanation

B.1.1 'Database' algorithm

The aim of the algorithm is to implement Hamiltonian simulation for $H = \sum_{\gamma=1}^{\Gamma} w_\gamma H_\gamma$, via 'Taylorization', that is, via a Taylor series:

$$e^{-iHt/r} \approx \tilde{U}_r := \sum_{k=0}^{K}\frac{(-iHt/r)^k}{k!} = \sum_{k=0}^{K}\sum_{\gamma_1,\ldots,\gamma_k=1}^{\Gamma}\frac{(-it/r)^k}{k!} w_{\gamma_1}\ldots w_{\gamma_k} H_{\gamma_1}\ldots H_{\gamma_k}, \quad (22)$$

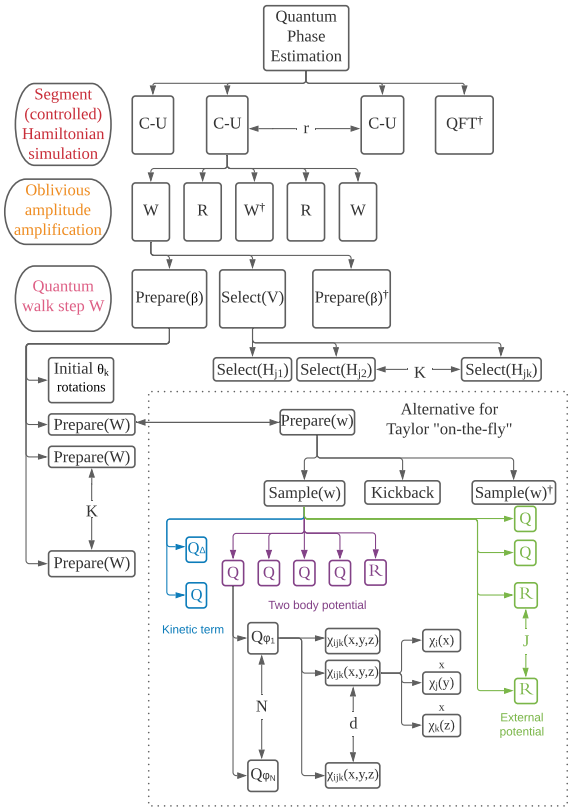

Figure 5: Abstraction level decomposition of the Taylor 'database' algorithm. The x-axis represents the time steps of the algorithm, while the y-axis is the abstraction level, higher meaning more abstract. In the lower box, we also depict the substitution one does to perform the alternative Taylor 'on-the-flight' algorithm. Notice that this does not show minor operations such as the computation of $\vec{\xi}$ or the multiplication in the last step of figure 4 from [3].

with $K = O\left(\frac{\log(r/\epsilon_{HS})}{\log\log(r/\epsilon_{HS})}\right)$. This means that in the Linear Combination of Unitaries formalism, we can write, $\tilde{U} = \sum_j \beta_j V_j$ with $\beta_j = \frac{t^k}{r^k k!} w_{\gamma_1} ... w_{\gamma_k}$ and $V_j = (-i)^k H_{\gamma_1} ... H_{\gamma_k}$.

Therefore we have to define how to implement Prepare(β) and Select(V), defined as

$$\text{Prepare}(\beta)\,|0\rangle^J = \sqrt{\frac{1}{s}} \sum_j \sqrt{\beta_j}\,|j\rangle \tag{23a}$$

depicted in figure 1 of [3], and

$$\text{Select}(V)\,|j\rangle\,|\psi\rangle = |j\rangle\,V_j\,|\psi\rangle\,. \tag{23b}$$

These operators use Prepare(W) and Select(H) respectively:

$$\text{Prepare}(W)\,|0\rangle^{\otimes\lceil\log_2\Gamma\rceil} = \sqrt{\frac{1}{\lambda}} \sum_{\gamma=1}^{\Gamma} \sqrt{w_\gamma}\,|\gamma\rangle \tag{24a}$$

with $\lambda = \sum_j |w_j| = O(N^4)$, and

$$\text{Select}(H)\,|\gamma\rangle\,|\psi\rangle = |\gamma\rangle\,H_\gamma\,|\psi\rangle\,, \tag{24b}$$

or in other words

$$\text{Select}(H)\,|ijkl\rangle\,|\psi\rangle = |ijkl\rangle\,a_i^\dagger a_j^\dagger a_k a_l\,|\psi\rangle\,. \tag{24c}$$

To implement (24c) we have to transform the creation and annihilation operators according to eq. 20 and 21 from [3]. This same article suggests introducing four additional qubits so that eq. 23 and 24 from [3] are finally used, containing only controlled Pauli operators.

Using those operators, we define the quantum walk step implementing $\tilde{U}_r$ (figure 2 in [3])

$$\mathcal{W} = (\text{Prepare}(\beta) \otimes \mathbf{1})^\dagger \text{Select}(V) (\text{Prepare}(\beta) \otimes \mathbf{1}) \tag{25a}$$

$$\mathcal{W}\,|0\rangle^J\,|\psi\rangle = \frac{1}{s}\,|0\rangle\,\tilde{U}_r\,|\psi\rangle + \sqrt{1 - \frac{1}{s^2}}\,|\Phi\rangle\,. \tag{25b}$$

To be able to use oblivious amplitude amplification, we need $s \approx 2$ [9], what can be achieved if $r = \lambda t / \ln 2$. Then $s = \sum_j |\beta_j| = \sum_{k=0}^{K} \frac{1}{k!} \ln 2^k \approx 2$.

B.1.2 'On-the-fly' algorithm

The main difference with the 'database algorithm' is that this algorithm aims to compute the integrals on-the-fly.

One starts observing that the Hamiltonian is constant in time, but at the same time it can be expressed as a spatial integral over a given region $\mathcal{Z}$, given that it decays exponentially outside it

$$H = \int_{\mathcal{Z}} \mathcal{H}(\vec{z}) d\vec{z} \approx \frac{\mathcal{V}}{\mu} \sum_{\rho=1}^{\mu} \mathcal{H}(\vec{z}). \tag{26}$$

As done in previous appendices, we divide the Hamiltonian evolution into segments U_r,

$$U_r \approx \sum_{k=0}^{K} \frac{(-it/r)^k}{k!} \int_{\mathcal{Z}} \mathcal{H}(\vec{z}_1)...\mathcal{H}(\vec{z}_k) d\vec{\boldsymbol{z}}. \tag{27}$$

If we substitute the integrals by Riemannian sums, $\mathcal{H}(\vec{z}) = \sum_{\gamma=1}^{\Gamma} w_\gamma(\vec{z}) H_\gamma$,

$$U_r \approx \sum_{k=0}^{K} \frac{(-it\mathcal{V})^k}{r^k \mu^k k!} \cdot \cdot \sum_{\gamma_1,...,\gamma_k=1}^{\Gamma} \sum_{\rho_1,...,\rho_k=1}^{\mu} w_{\gamma_1}(\vec{z}_{\rho_1})...w_{\gamma_k}(\vec{z}_{\rho_k}) H_{\gamma_1}...H_{\gamma_k} \tag{28}$$

Now, the question is how to prepare $w_{\gamma_i}(\vec{z}_{\rho_i})$ in the amplitudes. What the article does is first assume we have a method sample(w) such that

$$\text{sample}(w) |\gamma\rangle |\rho\rangle |0\rangle^{\otimes \lceil \log_2 M \rceil} = |\gamma\rangle |\rho\rangle |\tilde{w}_\gamma(\vec{z}_\rho)\rangle \tag{29}$$

with $\tilde{w}_\gamma(\vec{z}_\rho)$ an approximation of $w_\gamma(\vec{z}_\rho)$. Then the preparation procedure of the amplitudes consists of calculating the coefficients $w_{\gamma,m}(\vec{z}_\rho) \in \{\pm 1\}$ of a superposition such that $w_\gamma(\vec{z}) \approx \zeta \sum_{m=1}^{M} w_{\gamma,m}(\vec{z})$; $\zeta = \Theta\left(\frac{\epsilon_H}{\Gamma \mathcal{V} t}\right)$. To do that, defining $|l\rangle = |\gamma\rangle |m\rangle |\rho\rangle$, one performs *Kickback*:

$$|l\rangle |\tilde{w}_\gamma(\vec{z}_\rho)\rangle \to \begin{cases} |l\rangle |\tilde{w}_\gamma(\vec{z}_\rho)\rangle & \tilde{w}_\gamma(\vec{z}_\rho) > (2m - M)\zeta \\ i |l\rangle |\tilde{w}_\gamma(\vec{z}_\rho)\rangle & \tilde{w}_\gamma(\vec{z}_\rho) \leq (2m - M)\zeta \end{cases} \tag{30}$$

before uncomputing sample(w).

In summary, to prepare the amplitudes, one calculates sample(w) in the basis, performs (30) in a superposition of $|m\rangle$, and uncomputes the register prepared by sample(w). We will call such procedure Prepare(w):

$$\text{Prepare}(w) |0\rangle^{\otimes \lceil \log_2 L \rceil} = \sqrt{\frac{1}{\lambda'}} \sum_{l=1}^{L} \sqrt{\frac{\zeta \mathcal{V}}{\mu} w_{\gamma,m}(\vec{z}_\rho)} |l\rangle, \tag{31}$$

where $\lambda' = L\frac{\zeta \mathcal{V}}{\mu} = \Theta(\Gamma \mathcal{V} \max_{\vec{z},\gamma} |w_\gamma(\vec{z})|)$; $L = \Theta(\Gamma \mu M)$ and $M = \Theta(\max_{\vec{z},\gamma} |w_\gamma(\vec{z})|/\zeta)$. Additionally, due to equation 66 from [3] we know that

$$\mathcal{V} \max_{\vec{z},\gamma}(|w_\gamma(\vec{z})|) = 2^6 \varphi_{\max}^4 x_{\max}^5, \tag{32}$$

where the 2^6 is due to there being a hypercube with $(2x_{\max}/\delta x)^6$ terms.

This means that this alternative algorithm is similar to the 'database' one, but substitutes Prepare(W) with Prepare(w) that we just explained. The preparation over $|k\rangle$ is similar to the one depicted in figure 1 of [3], except that λ gets substituted by λ'.

The final, important detail we have to explain is how to perform the sample(w) routine. We want to calculate

$$\begin{aligned} w_\gamma(\vec{z}) = h_{ijkl}(\vec{x}, \vec{y}) &= \frac{\varphi_i^\dagger(\vec{x}) \varphi_j^\dagger(\vec{y}) \varphi_l(\vec{x}) \varphi_k(\vec{y})}{|\vec{x} - \vec{y}|} \\ &= \varphi_i^\dagger(\vec{x}) \varphi_j^\dagger(\vec{x} - \vec{\xi}) \varphi_l(\vec{x}) \varphi_k(\vec{x} - \vec{\xi}) |\vec{\xi}| \sin(\theta), \end{aligned} \tag{33a}$$

with $\vec{\xi} = \vec{x} - \vec{y}$ and θ the polar angle of $\vec{\xi}$; as well as

$$\begin{aligned} w_\gamma(\vec{z}) &= h_{ik}(\vec{x}) \\ &= \varphi_i^\dagger(\vec{x}) \left(-\sum_{j=0,1,2} \frac{\nabla_j^2}{2} - \sum_{j=0,...,J} \frac{Z_j}{|\vec{R}_j - \vec{x}|} \right) \varphi_k(\vec{x}) \\ &= -\varphi_i^\dagger(\vec{x}) \frac{\nabla^2}{2} \varphi_k(\vec{x}) \\ &\quad - \sum_j Z_j |\vec{\xi}_j| \sin(\theta_j) \varphi_i^\dagger(\vec{R}_j - \vec{\xi}_j) \varphi_k(\vec{R}_j - \vec{\xi}_j) \end{aligned} \tag{33b}$$

again transforming to polar coordinates in the external potential, $\vec{\xi}_j = \vec{R}_j - \vec{x}$. We need a subroutine Q to calculate the integrals.

$$\begin{aligned} Q &= \prod_{j=1}^{N} |j\rangle \langle j| \otimes Q_{\varphi_j}, \\ Q_{\varphi_j} |\rho\rangle |0\rangle^{\otimes \lceil \log_2 M \rceil} &= |\rho\rangle |\varphi_j(\vec{z}_\rho)\rangle. \end{aligned} \tag{34}$$

From the previous equation, one can see that the complexity of Q is N times the complexity of Q_{φ_j}. Notice that we will have to integrate over the space volume $\mathcal{V}$, summing over its discretization.

B.2 How to compute its cost

B.2.1 'Database' algorithm

We will use figure 5 as the main guide to compute the cost of the different abstraction levels. The first thing we have to take is the simulation time required, fixed by the error in the Phase Estimation algorithm, ϵ_{QPE}. One takes the number of segments $\tilde{U}_r$ to be

$$r = \frac{\lambda t}{\ln 2} = \frac{\pi \lambda}{\epsilon_{QPE} \ln 2}. \tag{35}$$

Another important parameter is the value of K, that controls the number of Prepare(W) in Prepare(β) and Select(H) in Select(V), which we can take from [45] to be

$$K = \left\lceil -1 + \frac{2 \log(2r/\epsilon_{HS})}{\log(\log(2r/\epsilon_{HS}) + 1)} \right\rceil. \tag{36}$$

The final aspects to take into account are:

1. θ_k **initial rotations**. This can be done using $K-1$ controlled R_y rotations.

2. **Prepare(W)** The cost of an arbitrary state preparation for can be estimated as $2^{\lceil \log_2 N^4 \rceil + 1}$ arbitrary rotations, using the protocol from [61], as it is preferable to encode $|ijkl\rangle$ instead of a continuous register that later on gets converted to that. This will be the most expensive part of the algorithm.

3. **Select(H)** First we have to specify how to create the circuit for each operator $a_{j,q}$ (analogously $a^\dagger_{j,q}$). For that we iterate over $n \in \{1, ..., N\}$. If $j = n$ we apply a σ_x or $\pm i\sigma_y$ as dictated by $|q\rangle$, if $j < n$ then we apply σ_z.

   The equality case can be performed via multi-controller Pauli operators. For each creation/annihilation operator, there will be $4N$ options due to the possible values of $|j\rangle\,|q\rangle$. We have to control on one qubit of register $|k\rangle$ encoded in unary to take into account the amplitude term corresponding to $\frac{(t/r)^k}{k!}$, on $|j\rangle$ with $\lceil \log_2 N \rceil$ qubits, and on $|q\rangle$; we will need to resort to multi controlled gate decomposition.

   To avoid the comparison in the case of $n < j$ we can create an accumulator. That is, when $n = j$ we switch an ancilla from $|1\rangle \to |0\rangle$, and controlled on such ancilla (and the unary register $|k\rangle$), at each step we perform σ_z on the $n-$th register of $|\psi\rangle$. This means N Toffolis and N multi-controlled (on $\lceil \log_2 N \rceil$ qubits) Not gates due to the equality comparison.

B.2.2 'On-the-fly' algorithm

To compute the cost of the 'on-the-fly' variation of this algorithm, the key step is substituting the Prepare(W) operator by something less expensive. The way we do this is by computing the one and two body integrals on the fly, by creating a sign-weighted superposition in register $|\rho\rangle$. Such superposition will use $\lceil \log_2 \mu \rceil$ qubits and can take values from 0 to $\mu - 1$ where

$$\begin{aligned}\mu &\approx \left(\frac{2r \times 6K}{\epsilon_H}(4\varphi'_{\max} + \varphi_{\max}/x_{\max})\varphi^3_{\max}x^6_{\max}\right)^6 \\ &= \Theta\left(\left(\frac{N^4 t}{\epsilon_H}(\varphi'_{\max} + \varphi_{\max}/x_{\max})\varphi^3_{\max}x^6_{\max}\right)^6\right)\end{aligned} \tag{37}$$

as can be seen from equations 73 and 74, and the text in the paragraph before equation 61, from [3]. Although this is a large number, it will only appear logarithmically in the number of qubits in the $|\rho\rangle$ register as explained in (28), so does not represent a too large complexity overhead. Notice that from equation 60 in [3], $r = \frac{\lambda' t}{\ln 2} = \frac{t}{\ln 2}\Gamma\mathcal{V}\max_{\vec{z},\gamma}(|w_\gamma(\vec{z})|)$, and the factor of 4 in front of $\varphi'_{\max}$ appear because we were deriving $\varphi^4_{\max}$; whereas the 2 appears because if we assume a hypercube, there should be $(2x_{\max}/\delta x_{\max})^6$ blocks in the discretization. Additionally, we can choose the coordinate system centered around the orbital such that $x_{\max} = O(\log(Nt/\epsilon_H)) = C\log(Nt/\epsilon_H)$, C a constant given by the software package users. $\varphi_{\max}$ will not depend on N. Similarly, since ζ is ϵ_H divided by the number of integral terms calculated in the process,

$$M = \frac{\max_{\vec{z},\gamma}(|w_\gamma(\vec{z})|)}{\zeta} = \frac{6Kr\Gamma\mathcal{V}\max_{\vec{z},\gamma}(|w_\gamma(\vec{z})|)}{\epsilon_H}, \tag{38}$$

where we can use the expressions from (32).

The final contribution we should take into account is that of the arithmetic operations required to calculate $\varphi_j(\vec{z}_\rho)$, which will also depend on the basis function we are using.

For that we will be using quantum addition [27], multiplication [52] and integer division [67]. The respective T-gate costs are $4n + O(1)$, $21n^2 - 14$ and $14n^2 + 7n + 7$, where n is the number of digits, $n = \lceil \log_2 \mu \rceil/3$, as there are three coordinates. Additionally, performing comparison between two numbers [20] can be done using $2n$ Toffoli gates if each of the inputs to compare is length n, so $8n$ T-gates.

To calculate the number of operations needed, we have to first remember that we are using a Gaussian basis set. In such basis, we expand the wave function as $\phi = \sum_{i=1}^M c_i\chi_i$. Each $\chi_j(x, y, z) = (x - X)^k(y - Y)^l(z - Z)^m e^{-\zeta_i(\mathbf{r}-\mathbf{R})^2}$, where (X, Y, Z) indicate the center of the atom, and $k + l + m$ is the angular momentum (eg. $k + l + m = 1$ means p-type basis etc. We assume that we only use up to d basis). The orbitals are usually contracted $\kappa_j = \sum_{i=1}^d d_{ij}\chi_i$ and $\phi = \sum_{j=1}^N c_j\kappa_j$. Each κ_j is one of the N basis functions that we use. More information on the topic of Gaussian basis sets might be found in a recent review [31].

In any case, to calculate each basis function $\kappa_j = \varphi_j$ we have to do the following:

1. Calculate $(x - X)$, $(y - Y)$, and $(z - Z)$, using $12n + O(1)$ T gates.

2. Calculate $(\mathbf{r}-\mathbf{R})^2 = (x-X)^2+(y-Y)^2+(z-Z)^2$, with cost $3(21n^2 - 14)$ for the multiplications, that is the leading cost. The sums mean $8n+O(1)$ additional cost.

3. Calculate the exponential $\zeta_i(\mathbf{r}-\mathbf{R})^2$ with a single multiplication, at T-gate cost $(21n^2 - 14)$.

4. $e^{-\zeta_i(\mathbf{r}-\mathbf{R})^2}$ via a Taylor series. Expanding to order o means $o - 1$ multiplications and divisions, and o sums.

5. The error in the previous expansion can be bounded as $\max\left(\zeta_i(\mathbf{r} - \mathbf{R})^2\right)^o/o!$

6. To construct $\chi_j(x, y, z)$ we need 3 multiplications, so the cost is $\approx 3(21n^2 - 14)$.

7. Each κ_j will be a sum of weighted exponentials, so the previous cost should be multiplied by d, the number of terms in such sum.

The number of terms d in each κ_j depends on the basis used, but it can be seen in tables 1-4 from [31] that the number of primitive basis sets χ_i that form each κ_j does not exceed 6 functions in the case of segmented basis sets (sparse d_{ij}), so we will take $d = 6$. However, if the basis set is general-contracted, d_{ij} is dense and the number might be much greater.

Once we have computed $\kappa_j = \varphi_j(\vec{x})$, we want to compute $\tilde{w}_\gamma(\vec{z})$:

- Whenever we have to compute $\vec{\xi}_j$ or $\vec{\xi}_j$, the cost is $12n + O(1)$ T-gates.

- Performing $\mathcal{R}|\vec{\xi}\rangle|0\rangle \mapsto |\vec{\xi}\rangle||\vec{\xi}|\sin\theta\rangle$, and similarly for $\vec{\xi}_j$. To do that, observe that $|\vec{\xi}|\sin\theta = \sqrt{\vec{x}_x^2 + \vec{x}_y^2}$, so we need two multiplications at cost $2(21n^2 - 14)$, one sum at T-gate cost $4n + O(1)$, and a square root calculation. We compute the square root using the Babylonian method, which only involves a sum and a division per order.

- $\nabla^2\chi_k(x) = \left(4x^2 - 2 + 4k - (1+k)/x^2\right)\chi_k(x)$. If we call the parenthesis $a_k(x)$, then $\nabla^2\chi_{ijk}(x,y,z) = (a_i(x) + a_j(y) + a_k(z))\chi_{ijk}$. Computing $a_i(x)$ can be done using 4 sums, 1 multiplication (x^2 term) and 1 division. This is because multiplying by 4 is free, just shifting bit positions. This has to be multiplied by 3 to take into account the three coordinates in the Laplacian, and done before the combination of the d functions into a single $\kappa_j = \varphi_j$.

In a similar fashion can Q_Δ be computed, for the sake of a name for outputting $\nabla^2\varphi$.

Overall, the cost of Sample(w) is

- Two-body term: 4Q $+\mathcal{R}$ + 4 multiplication + computation of $\vec{\xi}$.

- Kinetic term: $Q + Q_\Delta$ + multiplication.

- External potential term: 2Q $+J \times \mathcal{R}$ + J multiplications by Z_j and $J-1$ sums + J computations of $\vec{\xi}_j$.

Remember that in the previous calculations we are taking $n = \lceil \log_2 \mu \rceil/3$.

The cost of the rotation *Kickback* between the two applications of Sample(w) can be seen as a controlled rotation on the result of a comparison with $\lceil \log_2 \mu \rceil$ qubits. This requires one sum, one multiplication, and one comparison, which should be done twice to uncompute the result once the rotation has happened. From the previous, the cost of the 'on-the-fly' version of algorithm [3] can be computed using figure 5.

B.3 How to adapt the Hamiltonian simulation to control the direction of the time evolution

Quantum Phase Estimation requires being able to control the time direction of the Hamiltonian evolution of a segment. We do that by slightly modifying the Select(V) operator: if we want to simulate $e^{-iHt/r}$, for $k = 4j + 1$ we apply a C-S† operation (to apply $-i$ phase) and C-S if $k = 4j + 3$, while if we instead want to simulate $e^{iHt/r}$ additionally apply C-X in those situations to flip the sign. Here the Control bits are the value of k and the control qubits in Quantum Phase Estimation.

Adapting the Hamiltonian simulation method for Phase Estimation operation then amounts to two multi-controlled Not gates, with $K/2 + 1$ controls because k is encoded in unary and we are using Bayesian Phase Estimation with a single control ancilla.

C Configuration interaction and first quantization

C.1 Method explanation

In the previous section, we saw how to use Taylorization as a Hamiltonian simulation method in second quantization. Here, we explain the approach of [4], which relies on the same approach but in first quantization, in a formulation called Configuration Interaction. The general structure of the algorithm will consequently be similar.

In the Configuration Interaction representation one writes $|\alpha\rangle = |\alpha_0, ..., \alpha_{\eta-1}\rangle$, where each α_i indicates an occupied orbital. The determinant of the corresponding wave functions is an antisymmetric function called Slater determinant and represents the state of the system

$$\langle \vec{r}_0, ...\vec{r}_{\eta-1}|\alpha\rangle = \frac{1}{\sqrt{\eta!}}\left|\begin{pmatrix} \varphi_{\alpha_0}(\vec{r}_0) & \cdots & \varphi_{\alpha_{\eta-1}}(\vec{r}_0) \\ \vdots & & \vdots \\ \varphi_{\alpha_0}(\vec{r}_{\eta-1}) & \cdots & \varphi_{\alpha_{\eta-1}}(\vec{r}_{\eta-1}) \end{pmatrix}\right|. \tag{39}$$

An important aspect of this method is that it can only be applied with local basis functions, such as Gaussian orbitals, but not the plane-wave basis. The reason is that at one point one has to bound the error by approximating Hamiltonian integrals from Riemannian sums, and bounding the error is only possible if we are restricted to a local volume of space. To make it work with molecular orbitals appearing in the Hartree-Fock procedure, one can use the operator $U = \exp\left(-\sum_{ij}\kappa_{ij}a_i^\dagger a_j\right)$ that changes the basis and may be applied using $\tilde{O}(N^2)$ gates [71]. κ here is an antihermitian matrix that is obtained by the self-consistent Hartree Fock procedure.

Expressing the Configuration Interaction Hamiltonian as a linear combination of unitaries is not efficient. On the other hand, though, it can be expressed as a sparse matrix, called Configuration Interaction (CI), whose elements are a sum of integrals.

The Slater-Condon rules indicate how to compute those matrix elements, based on one- and two-body integrals [4]. Because of them, the sparsity of the Configuration Interaction matrix is

$$\begin{aligned} d &= \binom{\eta}{2}\binom{N-\eta}{2} + \binom{\eta}{1}\binom{N-\eta}{1} + 1 \\ &= \frac{\eta^4}{4} - \frac{\eta^3 N}{2} + \frac{\eta^2 N^2}{2} + O(\eta^2 N + \eta N^2) \in O(\eta^2 N^2). \end{aligned} \tag{40}$$

After decomposing the Configuration Interaction matrix in 1-sparse operators, we approximate its integrals as a Riemannian sum of self inverse operators. Finally, we construct Select($\mathcal{H}$), that applies such self inverse operators

$$\text{Select}(\mathcal{H}) \left|l\right\rangle \left|\rho\right\rangle \left|\psi\right\rangle = \left|l\right\rangle \left|\rho\right\rangle \mathcal{H}_{l,\rho} \left|\psi\right\rangle \tag{41}$$

and allows to evolve the system under the Hamiltonian. The steps are the following:

1. **Decompose the Hamiltonian into 1-sparse operators**. Such operators will be indexed by 2 4-tuples (a_1, b_1, i, p) and (a_2, b_2, j, q) that denote the differing orbitals. This tuples will be used to perform the operator

$$Q^{col} : \left|\gamma\right\rangle \left|\alpha\right\rangle \left|0\right\rangle^{\eta \lceil \log_2 N \rceil} \mapsto \left|\gamma\right\rangle \left|\alpha\right\rangle \left|\beta\right\rangle, \tag{42}$$

   within the Select operator (41). The specific algorithms for this procedure can be found in appendix A of the article of reference for this appendix [4]. These procedures require, between other things, the ability to order a list of orbitals, which we explain in Algorithm 1.

2. **Decompose each 1-sparse operator into h_{ij} and h_{ijkl}**. The Slater Condon rules sometimes requires the sum over η integrals. Here we decompose the previous sum such that only at most two integrals are summed for each term. This decomposition can be seen in section 4.2 of the original article [4]. It will allow us to write the Hamiltonian as $H = \sum_\gamma H_\gamma$, with $\Gamma = \eta + \eta(\eta-1)/2 + (N-1)\eta^2 + (N-1)^2\eta(\eta-1)/2$.

3. **Discretising the integrals into Riemannian sums**.

   Each Hamiltonian term from the previous equation might be represented as $H_\gamma^{\alpha\beta} = \int \aleph_\gamma^{\alpha\beta}(\vec{z})d\vec{z}$. Since the domain of each integral might be different, we write $H_\gamma^{\alpha\beta} \approx \sum_{\rho=1}^{\mu} \aleph_{\gamma\rho}^{\alpha\beta}$. Here is where we need the requirement that the orbitals are local.

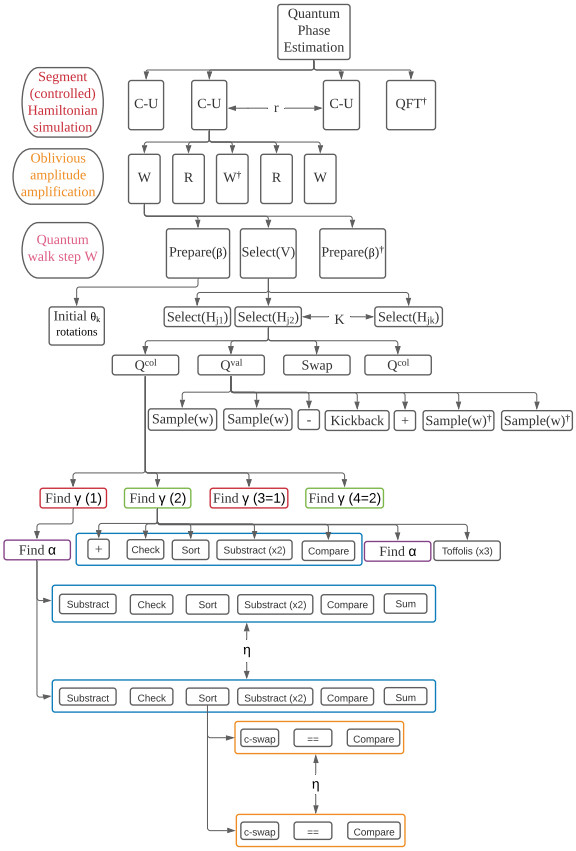

Figure 6: Abstraction level decomposition of the Configuration Interaction procedure [4]. The Sample operation shown is the same as in figure 5.

4. **Decomposition into self-inverse operators**. Finally, we decompose in a sum of $M \in \Theta(\max_{\gamma,\rho}||\aleph_{\gamma,\rho}||_{\max}/\zeta)$ self-inverse operators, using a similar strategy as in the previous section B [3]. Operators will be indexed by ρ and $l = (\gamma, m, s)$, where m controls whether a phase i is added in the Kickback, and s is sign. ρ controls the Riemmanian sum. The final decomposition can be written as $H = \zeta \sum_{l=1}^{L} \sum_{\rho=1}^{\mu} \mathcal{H}_{l,\rho}$. Using this we can perform

$$Q^{val} |l\rangle |\rho\rangle |\alpha\rangle |\beta\rangle = \mathcal{H}_{l,\rho}^{\alpha\beta} |l\rangle |\rho\rangle |\alpha\rangle |\beta\rangle , \quad (43)$$

which also appears in the Select operator.

In conclusion, one time segment of the Taylorized Hamiltonian evolution will be

$$U_r \approx \sum_{k=0}^{K} \frac{(-it\zeta)^k}{r^k k!} \sum_{l_1,\dots,l_k=0}^{L} \sum_{\rho_1,\dots,\rho_k=0}^{\mu} \mathcal{H}_{l_1,\rho_1} \dots \mathcal{H}_{l_k,\rho_k}, \quad (44)$$

where $|l\rangle = |\gamma, m, s\rangle$. The role of Prepare will be restricted to the preparation of θ angles for $\frac{(-it\zeta)^k}{r^k k!}$.

To compute the algorithm cost, we will need constants α, γ_1 and γ_2 to comply with equations 28, 29 and 30 from [4], and will bound the error from computing the Hamiltonian integrals as Riemannian sums:

- For each l there is a vector c_l such that if $||\vec{r} - \vec{c}_l|| \geq x_{\max}$ then

$$|\varphi_l(\vec{r})| \leq \varphi_{\max} \exp\left(-\frac{\alpha}{x_{\max}} ||\vec{r} - \vec{c}_l||\right) \quad (45)$$

- For each l, φ_l is twice differentiable and there exists γ_1 and γ_2 such that

$$||\nabla \varphi_l(\vec{r})|| \leq \gamma_1 \frac{\varphi_{\max}}{x_{\max}} \quad (46a)$$

and

$$||\nabla^2 \varphi_l(\vec{r})|| \leq \gamma_2 \frac{\varphi_{\max}}{x_{\max}^2} \quad (46b)$$

C.2 How to compute its cost

We will use figure 6 as a guide to computing the cost of the algorithm. There are three key differences with the cost calculated in the previous appendix. First, some parameters change. These are notably r, the number of time segments, and M, which indicates the size of register $|m\rangle$ and as a consequence influences the cost. The other two aspects that change are that we need to compute the cost of Q^{val} and Q^{col} in figure 6.

Let us start computing r, the number of segments. $r = \zeta L \mu t / \ln(2)$ (according to the paragraph before equation 68 in [4]), with $L = 2(M\Gamma)$ (the 2 because of register s in $|l\rangle = |\gamma\rangle |m\rangle |s\rangle$). The product $\mu \max_{\gamma,\rho} ||\aleph_{\rho,\gamma}|| = \mu M \zeta$ can optimized from Lemmas 1-3 in the original article [4], so

$$r = 2\Gamma t (\mu M \zeta) / \ln(2), \quad (47)$$

with $t = \pi/\epsilon_{QPE}$ and

$$\Gamma = \binom{\eta}{2}\binom{N-\eta}{2} + \binom{\eta}{1}\binom{N-\eta}{1} + 1 = \frac{\eta^4}{4} - \frac{\eta^3 N}{2} + \frac{\eta^2 N^2}{2} + O(\eta^2 N + \eta N^2) \in O(\eta^2 N^2). \quad (48)$$

To compute M, similarly as in the previous appendix

$$M = \Theta\left(\frac{\max_{\gamma,\rho} ||\aleph_{\rho,\gamma}||}{\zeta}\right), \quad (49)$$

and in the previous appendix we saw that ζ is the error that we allow, modelled as the error budget for this error source ϵ_H, divided by the number of times we called the decomposition, $\Gamma \mathcal{V} r$. The reason why $\mathcal{V}$ appeared in place of μ is because instead of writing

$$H_\gamma = \sum_\rho w_\gamma(\vec{z}_\rho) \quad (50a)$$

we were taking

$$H_\gamma = \frac{\mathcal{V}}{\mu} \sum_\rho w_\gamma(\vec{z}_\rho), \quad (50b)$$

so the precision must be scaled correspondingly. In this case however,

$$H_\gamma = \sum_\rho \aleph_\gamma(\vec{z}_\rho), \quad (51)$$

integrating the cell volume as a multiplicative constant in $\aleph_\gamma(\vec{z}_\rho)$, so the error has to be appropriately scaled by $\mathcal{V}/\mu$. Similarly, this time,

$$\zeta = \frac{\epsilon_H}{3 \cdot 2Kr(\#\gamma)(\#\rho)} = \frac{\epsilon_H}{6Kr\Gamma\mu}. \quad (52)$$

Since $\max_{\gamma,\rho} ||\aleph_{\rho,\gamma}||$ is bounded from Lemmas 1, 2 and 3 in [4], we can compute M. These lemmas will also depend on δ, taken to be the individual error in each of the integrals. Therefore, we should take (see paragraph before eq. 74 in [4]):

$$\delta = \frac{\epsilon_H}{6Kr}, \quad \zeta = \frac{\delta}{\Gamma\mu} \quad (53)$$

where $6K$ is the number of times these integrals are used in each segment, indicated figure 6. We can see that δ depends on r, which depends on $\mu M \zeta$, which from the previously mentioned lemmas depends on δ. We solve this by computing r such that μ times equations 39, 43 and 47 in [4] become approximate equalities to $\mu M \zeta$. This way we obtain a close result to if we had used $\delta = \epsilon_H/(6K\Gamma t)$.

Now let us turn to two main operators involved in the algorithm, Q^{val} in (43) and Q^{col} in (42). To compute the cost of Q^{val} the procedure is the same as we did in the previous appendix B. In this case, however, we will have to compute up to 2 basis functions. To do so we iterate over the different possibilities of γ to decompose in h_{ij} and h_{ijkl}.

a. $p = 0 = q$. This point requires calculating η terms of type $h_{\chi_i\chi_i}$, and $\eta(\eta-1)/2$ terms $(h_{\chi_i\chi_j\chi_i\chi_j} - h_{\chi_i\chi_j\chi_j\chi_i})$.

b. $p = 0$, $q \neq 0$. In this case there are $(N-1)\eta(\eta-1)$. terms of the form $h_{k\chi_i l\chi_i} - h_{k\chi_i\chi_i l}$, and $(N-1)\eta$ for the terms of the form h_{kl}.

c. $p \neq 0$, $q = 0$. No integrals are needed.

d. $p \neq 0$, $q \neq 0$. All of the integrals in this last point are of the form $h_{ijkl} - h_{ijlk}$. There are $(N-1)^2\eta(\eta-1)/2$ of them.

From this and the previous appendix B, the cost of Q^{val} can be readily calculated.

Computing Q^{col} requires implementing the procedure 'Find Alphas' and a more general one indicated in cases 2 and 4 in appendix A, that we will call 'Find Gammas' [4]. Both 'Find Alphas' and 'Find Gammas' require a sorting algorithm that has the peculiarity that only up to one item might be out of order, and we know its position. For that reason, we have described a possible sorting algorithm 1. To compute the cost, one should also make use of the basic operations described in table 2.

Algorithm1 Algorithm to order the orbitals $|\tilde{\alpha}\rangle$ generated from $|\beta\rangle$, shift $|p\rangle$ and position $|j\rangle$

procedure ORDER($|\beta\rangle\,|p\rangle\,|j\rangle$)
  Calculate unordered $|\tilde{\alpha}\rangle_1$ subtracting $|p\rangle$ from $|\beta_j\rangle$.
  Use Cnots to create two 'basis' copies of $|\tilde{\alpha}\rangle_1$, called $|\tilde{\alpha}\rangle_1$ and $|\tilde{\alpha}\rangle_2$
  for $i \in$ reversed(range(j)) **do**
    if then $|\tilde{\alpha}_i\rangle_1 == |\tilde{\alpha}_{i+1}\rangle_1$ then
      return Invalid ▷ If this is activated, reverse the entire computation. Thus cost $\times 2$.
    $|0\rangle_a \leftarrow (|\tilde{\alpha}_i\rangle_1 > |\tilde{\alpha}_j\rangle_1)$
    Controlled on $|\cdot\rangle_a$ swap $|\tilde{\alpha}_i\rangle_2$ and $|\tilde{\alpha}_{i+1}\rangle_2$
    Uncompute $|\cdot\rangle_a$
  Uncompute $|\tilde{\alpha}\rangle_1$
  return $|\beta\rangle\,|p\rangle\,|j\rangle\,|\tilde{\alpha}\rangle_2$

Using this and figure 6 it is relatively straightforward to compute the cost of the present algorithm. Notice however that the initial Hartree-Fock rotation $U = \exp\left(-\sum_{ij}\kappa_{kj}a_i^\dagger a_j\right)$ has not yet been implemented in the cost estimation, but it is not a dominant factor.

C.3 How to adapt the Hamiltonian simulation to control the direction of the time evolution

Adapting the Hamiltonian simulation for its use in Quantum Phase Estimation can be done as in appendix B.3. The cost can be therefore calculated in the same way.

D Introducing the QROM

D.1 Method explanation

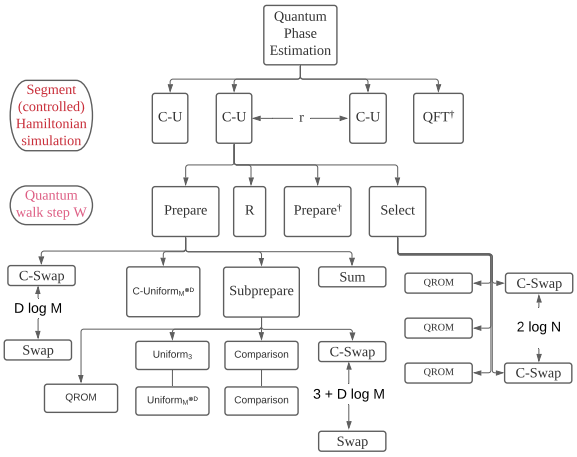

Figure 7: Abstraction level decomposition of the procedure [5].

One of the key innovations used in this method is that if instead of simulating $\mathcal{W}(H) = e^{\pm iH\tau}$ one chooses $\mathcal{W}(H) = e^{\pm i\arccos(H/\lambda)}$, one can eliminate the Taylor series error completely [5], as we already explained in section 3.3. This idea had been previously introduced [10, 56], and has the consequence that instead of phase estimating the ground state energy E_0 one phase estimates $\arccos(E_0)$. We can simulate $\mathcal{W}(H)$ with the standard quantum walk $(\text{Prepare}^\dagger \otimes \mathbf{1})\text{Select}(\text{Prepare}^\dagger \otimes \mathbf{1})$. Notice that in contrast to [62] we are using arccos instead of arcsin because, since $\arccos\theta + \arcsin\theta = \pi/2$, the change amounts to a global phase and sign change, and we want to use similar notation everywhere.

Therefore, in this appendix, we aim to explain the implementations of the Prepare and Select operators, and the key innovation of article [5], the proposal of an efficient QROM that will play an important role in both Prepare and Select. We will start with the latter. The role of the QROM is to iterate over all possible inputs preparing the corresponding outputs.

How can we construct such a unary iterator? The easiest way is just to use as control all the index qubits for each of the L values the indices can take. But this is clearly wasteful since we are often repeating the same controls over consecutive values in the indices. Therefore, [5] proposes using auxiliary qubits to hierarchically save the combinations of controls, giving rise to circuits similar to their figure 5, called the "sawtooth" circuit. This circuit, in contrast to the original, can be simplified avoiding the wasteful repetition of AND gates that we indicated previously. As shown in their figure 6, allows for converting their figure 5 to their figure 7, requiring only $(L-1)$ AND

gates. Since each AND can be constructed from 4 T gates, the unary iterator requires $4L-4$ T gates.

A variation over the previous iterator is the accumulator. Instead of directly applying the chosen gates to the target qubits, one defines an accumulator qubit, which is at state $|0\rangle$ until we control on the selected value of the indices and stays $|1\rangle$ until the end of the iterator, at which point it can be uncomputed since at the end the accumulator will be at disentangled state $|1\rangle$. A picture of this variant is figure 8 in [5]. This accumulator is specially useful because it will allow us to apply the Majorana fermion operator $|l\rangle|\psi\rangle \to |l\rangle\left(\frac{a_l^\dagger - a_l}{i}|\psi\rangle\right) = |l\rangle Y_l Z_{l-1}...Z_0|\psi\rangle$, as can be seen in figure 9 in [5].

This QROM is useful to perform the Prepare circuit. However, we will not prepare

$$|0\rangle^{\lceil \log_2 \Gamma\rceil} \mapsto \sum_{\gamma=0}^{\Gamma-1}\sqrt{\frac{w_\gamma}{\lambda}}, \tag{54}$$

but rather

$$|\mathcal{L}\rangle = \sum_{\gamma=0}^{\Gamma-1}\sqrt{\frac{w_\gamma}{\lambda}}|\gamma\rangle|\text{temp}_\gamma\rangle, \tag{55}$$

with $|\text{temp}_\gamma\rangle$ a junk register entangled with $|\gamma\rangle$. The way to ensure that this entanglement does not interfere with other computations is to ensure that the same qubits are fed into the uncomputation and that the reflection $\mathcal{R}_\mathcal{L} = (2|\mathcal{L}\rangle\langle\mathcal{L}| - \mathbf{1})$ that appears in the quantum walk step $\mathcal{W} = \mathcal{R}_\mathcal{L}\cdot\text{Select}$ is done only over state $|0\rangle$. Here, we will be looking for an algorithm that performs the following transformation:

$$|0\rangle^{\otimes(1+2\mu+2\lceil\log_2\Gamma\rceil)} \to \sum_{\gamma}^{\Gamma-1}\sqrt{\tilde{\rho}_l}|\gamma\rangle|\text{temp}_\gamma\rangle, \tag{56}$$

with $\tilde{\rho}_\gamma$ a μ-bits binary approximation to w_γ/λ. For this, one chooses

$$\mu = \left\lceil \log_2\left(\frac{2\sqrt{2}\lambda}{\Delta E}\right) + \log_2\left(1+\frac{\Delta E^2}{8\lambda^2}\right) - \log_2\left(1-\frac{||H||}{\lambda}\right)\right\rceil. \tag{57}$$

as given in equation 36 from [5]. Since the Hamiltonian is frustrated, the quotient in the last logarithm is upper bounded away from 1, and thus the last term is $O(1)$. Similarly, since $\Delta E < \lambda$, the second term can be upper bounded by $\log_2(1+1/8)$.

We will prepare this new $|\mathcal{L}\rangle$ indirectly, using a circuit that they depicted in figure 11 and called Subprepare. We start from the uniform superposition $\sum|\gamma\rangle$ and have two registers that depend on γ, $|\text{keep}_\gamma\rangle$ and $|\text{alt}_\gamma\rangle$. $|\text{keep}_\gamma\rangle$ will dictate the probability that we coherently exchange $|\gamma\rangle$ and $|\text{alt}_\gamma\rangle$. The objective is to find keep_γ and alt_γ such that in the end, we obtain the correct amplitudes. The details of the procedure can be found in section 3D in the main reference for this appendix, and it is the inverse procedure of the depicted one in their figure 13 [5].

The Hamiltonian basis explored in this technique is plane waves, with the same structure that we saw in eq. (67) [5]. The article suggests that to make the basis set as compact as possible, one may choose Gausslet basis sets, that combine some of the features of plane waves and of Gaussian waves [72, 73]. They represent however a very complex basis set, so for the time being we have not implemented it yet, working in dual waves instead.

The following question we need to answer is how to index the terms of the Hamiltonian. We will have registers $|p\rangle$ and $|q\rangle$ which in binary encode the orbitals without taking into account the spin, while $|\alpha\rangle$, and $|\beta\rangle$ will take that into account. Thus, $|p\rangle$ and $|q\rangle$ will encode numbers from 0 to $N/2-1$ (N the number of spin-orbitals) and will need $\lceil\log_2 N\rceil - 1$ qubits each. Then we will have two one-qubit registers $|U\rangle$ and $|V\rangle$, that will decide what term in the Hamiltonian to apply. Finally $|\theta\rangle$ will be used to apply a phase $(-1)^\theta$. Overall, we have the following Select operator

$$\begin{aligned}&\text{Select}\,|\theta,U,V,p,\alpha,q,\beta\rangle|\psi\rangle = \\ &(-1)^\theta|\theta,U,V,p,\alpha,q,\beta\rangle \\ &\otimes\begin{cases} Z_{p,\alpha} & U\wedge\neg V\wedge((p,\alpha)=(q,\beta)) \\ Z_{p,\alpha}Z_{q,\beta} & \neg U\wedge V\wedge((p,\alpha)\neq(q,\beta)) \\ X_{p,\alpha}\vec{Z}X_{q,\alpha} & \neg U\wedge\neg V\wedge(p<q)\wedge(\alpha=\beta) \\ Y_{p,\alpha}\vec{Z}Y_{q,\alpha} & \neg U\wedge\neg V\wedge(p>q)\wedge(\alpha=\beta) \\ \text{Undefined} & \text{Otherwise}\end{cases}.\end{aligned} \tag{58}$$

As an aside notice that p and q are three dimensional vectors whose elements take integer values in the range $[0,(N/2)^{1/3}-1]$, so we need to map (p,σ) to an integer index representing a qubit. The mapping is, for a D dimensional system ($D=3$)

$$M = (N/2)^{1/D}, \qquad f(p,\sigma) = \delta_{\sigma,\uparrow}M^D + \sum_{j=0}^{D-1}p_j M^j. \tag{59}$$

Similarly, the Prepare operator performs

$$\begin{aligned}&\text{Prepare}: |0\rangle^{\otimes(3+2\lceil\log_2 N\rceil)} \mapsto \\ &\sum_{p,\sigma}\tilde{U}(p)|\theta_p\rangle|1\rangle_U|0\rangle_V|p,\sigma,p,\sigma\rangle \\ &+\sum_{p\neq q,\sigma}\tilde{T}(p-q)|\theta^{(0)}_{p-q}\rangle|0\rangle_U|0\rangle_V|p,\sigma,q,\sigma\rangle \\ &+\sum_{(p,\alpha)\neq(q,\beta)}\tilde{V}(p-q)|\theta^{(1)}_{p-q}\rangle|0\rangle_U|0\rangle_V|p,\sigma,q,\sigma\rangle,\end{aligned} \tag{60}$$

with coefficients

$$\tilde{U}(p) = \sqrt{\frac{|T(0) + U(p) + \sum_q V(p-q)|}{2\lambda}}$$
$$\tilde{T}(p) = \sqrt{\frac{|T(p)|}{\lambda}}; \qquad \tilde{V}(p) = \sqrt{\frac{|V(p)|}{4\lambda}} \tag{61}$$

and

$$\theta_p = \frac{1 - sign(-T(0) - U(p) - \sum_q V(p-q))}{2}$$
$$\theta_p^{(0)} = \frac{1 - sign(T(p))}{2}; \qquad \theta_p^{(1)} = \frac{1 - sign(V(p))}{2}. \tag{62}$$

To implement Prepare, first, we prepare a unitary operator called Subprepare, which acts as

$$|0\rangle^{\otimes(2+\log_2 N)} \mapsto$$
$$\sum_{d=0}^{N-1} \Big(\tilde{U}(d)\,|\theta_d\rangle\,|1\rangle_U\,|0\rangle_T + \tilde{T}(d)\,|\theta_d^{(0)}\rangle\,|0\rangle_U\,|0\rangle_V \tag{63}$$
$$+ \tilde{V}(d)\,|\theta_d^{(1)}\rangle\,|0\rangle_U\,|1\rangle_V\Big)\,|d\rangle\,.$$

The construction of Select, Subprepare and Prepare can be seen in fig. 14, 15 and 16 from [5]. Taking this into account, the total cost will be $r(2 \cdot \text{Prepare} + \text{Select} + R)$, where R stands for the reflection in each step.

D.2 How to compute its cost

The circuit implementing the Select operator is depicted in the above-mentioned figure 14 [5]. It will require the use of 3 QROM applications of size $O(N)$, and $2\lceil\log_2 N\rceil$ controlled swaps (Fredking gates) each requiring one T gate. So, the total T-gate cost is $12N + 8\lceil\log_2 N\rceil - 14$.

Subprepare is the main building block for Prepare, and it is depicted in figure 15 in [5]. It uses one QROM, with AND complexity $3M^D - 1 = 3N/2 - 1$, so T complexity $6N - 4$. The 3 is due to the three possible combinations that can appear in $|U\rangle$ and $|V\rangle$, whereas M^D is due to register $|p\rangle$ having $D\lceil\log_2 M\rceil = \lceil\log_2 N/2\rceil$ qubits. Appart from the QROM, Subprepare contains $3 + \lceil\log_2 N/2\rceil = 2 + \lceil\log_2 N\rceil$ controlled swaps (each requiring a Toffoli gate or 4 T gates); two comparison test between 2 μ-sized registers; and finally operators $\text{Uniform}_M^{\otimes D}$ and Uniform_3.

The Uniform operators prepare an uniform superposition over the first L basis states, and is analyzed in figure 12 in [5]. Since in particular we are using Uniform_3 and $\text{Uniform}_M^{\otimes D}$, this will require $8\lceil\log_2 L\rceil + O(\log_2 \epsilon_{SS}^{-1}) = 8\lceil\log_2 3\rceil + O(\log \epsilon_{SS}^{-1})$ T gates in the first case, and $8D\log M + O(\log \epsilon_{SS}^{-1}) = 8\lceil\log_2 N\rceil - 8 + O(\log \epsilon_{SS}^{-1})$ in the second. The $O(\log \epsilon_{SS}^{-1})$ term stands for 2 rotations R_z in each Uniform operator. Overall, Subprepare requires $6N + 12\lceil\log_2 N\rceil + 10\mu + 16\lceil\log_2 \epsilon_{SS}^{-1}\rceil$ T gates.

The Prepare operator can be seen in figure 16 in [5]. It requires another $\text{Uniform}_M^{\otimes D}$, at cost $8\lceil\log_2 N\rceil + 8\lceil\log_2 \epsilon_{SS}^{-1}\rceil$; $D\lceil\log_2 M\rceil = \lceil\log_2 N\rceil - 1$ swaps with 4 times as many T gates; 2 multicontrolled Not gates with $\lceil\log_2 N\rceil$ controls each, which can be implemented using $16\lceil\log_2 N\rceil$ T gates [8]; and one sum over $D\lceil\log_2 M\rceil$ qubits.

With the previous, we have everything we need to calculate the total T gate cost accurately.

E Plane and dual wave basis

E.1 Method explanation

When looking for a basis of functions to perform chemical calculations, one is primarily looking for a basis that [6]

1. Leads to a small number of terms in the Hamiltonian.

2. Allows for simple preparation of initial state.

On the Gaussian basis, initial states are easy to prepare using the Hartree-Fock procedure. However, the Hamiltonian may have up to $O(N^4)$ terms.

One idea to avoid having so many terms in the Hamiltonian is to use the plane waves and dual wave basis. The plane wave basis functions have the form

$$\varphi_{\boldsymbol{\nu}}(\boldsymbol{r}) = \sqrt{\frac{1}{\Omega}} e^{i\boldsymbol{k}_{\boldsymbol{\nu}}\cdot\boldsymbol{r}}, \quad \boldsymbol{k}_{\boldsymbol{\nu}} = \frac{2\pi\boldsymbol{\nu}}{\Omega^{1/3}}, \qquad \boldsymbol{\nu} \in [-N^{-1/3}, N^{1/3}]^3 \in \mathbb{Z}^3. \tag{64}$$

In the plane wave basis, the Hamiltonian will take the form [6]

$$H = \underbrace{+\frac{2\pi}{\Omega} \sum_{\substack{(p,\sigma)\neq(q,\sigma') \\ \nu\neq 0}} \frac{c_{p,\sigma}^\dagger c_{q,\sigma'}^\dagger c_{q+\nu,\sigma'} c_{p-\nu,\sigma}}{k_\nu^2}}_{V}$$
$$\underbrace{+\frac{1}{2}\sum_{p,\sigma} k_p^2 c_{p,\sigma}^\dagger c_{p,\sigma}}_{T} \underbrace{-\frac{4\pi}{\Omega} \sum_{\substack{p\neq q; \\ j,\sigma}} \left(\zeta_j \frac{e^{ik_{q-p}\cdot R_j}}{k_{p-q}^2}\right) c_{p,\sigma}^\dagger c_{q,\sigma}}_{U}, \tag{65}$$

$p, q \in [-N^{-1/3}, N^{1/3}]^3$ indexing the momentum. Notice that in this basis the kinetic operator T is diagonal, a property that we will use abundantly.

Fourier transforming (65), we get the dual plane

wave Hamiltonian,

$$H = \underbrace{\frac{1}{2N}\sum_{p,q,\nu,\sigma} k_\nu^2 \cos[k_\nu \cdot r_{q-p}] a^\dagger_{p,\sigma} a_{q,\sigma}}_{T} \underbrace{-\frac{4\pi}{\Omega}\sum_{p,j,\sigma,\nu\neq 0}\left(\frac{\zeta_j \cos[k_\nu \cdot (R_j - r_p)]}{k_\nu^2}\right) n_{p,\sigma}}_{U} + \underbrace{\frac{2\pi}{\Omega}\sum_{(p,\sigma)\neq(q,\sigma');\nu\neq 0}\frac{\cos[k_\nu \cdot r_{p-q}]}{k_\nu^2} n_{p,\sigma} n_{q,\sigma'}}_{V}, \tag{66}$$

with $n_p = a_p^\dagger a_p$, a_p and $a_p^\dagger$ the Fourier transformed annihilation and creation operators, and $\boldsymbol{r_p} = \boldsymbol{p}(\Omega/N)^{1/3}$. We can see that in this basis the potential terms become diagonal, and since the term V only has $\Theta(N^2)$ terms, the number of terms in the Hamiltonian is $O(N^2)$.

In Jordan Wigner mapping, (66) can be represented as

$$\begin{aligned} H = {} & \frac{\pi}{2\Omega}\sum_{\substack{(p,\sigma)\neq(q,\sigma')\\ \nu\neq 0}} \frac{\cos[k_\nu \cdot r_{p-q}]}{k_\nu^2} Z_{p,\sigma} Z_{q,\sigma'} \\ & \sum_{\substack{p,\sigma\\ \nu\neq 0}}\left(\frac{\pi}{\Omega k_\nu^2} - \frac{k_\nu^2}{4N} + \frac{2\pi}{\Omega}\sum_j \frac{\zeta_j \cos[k_\nu \cdot (R_j - r_p)]}{k_\nu^2}\right) Z_{p,\sigma} \\ & + \frac{1}{4N}\sum_{\substack{p\neq q\\ \nu,\sigma}} k_\nu^2 \cos[k_\nu \cdot r_{q-p}](X_{p,\sigma} Z_{p+1,\sigma} ... Z_{q-1,\sigma} X_{q,\sigma} \\ & + Y_{p,\sigma} Z_{p+1,\sigma} ... Z_{q-1,\sigma} Y_{q,\sigma}) + \sum_{\nu\neq 0}\left(\frac{k_\nu^2}{2} - \frac{\pi N}{\Omega k_\nu^2}\right) I. \end{aligned} \tag{67}$$

Depending on the situation, to simulate the Hamiltonian in the most efficient way possible we will jump back and forth between dual and primal representations depending on the operator of the Hamiltonian

$$H = \underbrace{FFFT^\dagger\left(\frac{1}{2}\sum_{\nu,\sigma} k_\nu^2 a^\dagger_{\nu,\sigma} a_{\nu,\sigma}\right) FFFT}_{T} \underbrace{-\frac{4\pi}{\Omega}\sum_{p,j,\sigma,\nu\neq 0}\left(\frac{\zeta_j \cos[k_\nu \cdot (R_j - r_p)]}{k_\nu^2}\right) n_{p,\sigma}}_{U} + \underbrace{\frac{2\pi}{\Omega}\sum_{(p,\sigma)\neq(q,\sigma');\nu\neq 0}\frac{\cos[k_\nu \cdot r_{p-q}]}{k_\nu^2} n_{p,\sigma} n_{q,\sigma'}}_{V} \tag{68}$$

where all the terms are diagonal. To implement this Hamiltonian, we need to perform a Fermionic Fast Fourier Transform (FFFT) [23], an adaptation of the classical Fast Fourier Transform. We cannot use here the Quantum Fourier Transform because we are using the Jordan-Wigner mapping that encodes the value of the qubits not in the amplitudes but the basis.

E.1.1 Trotterization algorithm

The most basic way to use the plane wave approach is to use (68) to simulate a segment of the Hamiltonian simulation procedure

$$\begin{aligned} e^{-iH\delta_t} \approx {} & e^{-i(U+V)\delta_t/2} \cdot \\ & FFFT^\dagger e^{-i(\delta_t/2)\sum_{\nu,\sigma} k_\nu^2 a^\dagger_{\nu,\sigma} a_{\nu,\sigma}} FFFT \\ & \cdot e^{-i(U+V)\delta_t/2} + O(\delta_t^3), \end{aligned} \tag{69}$$

with U and V given in (68). This formulation allows us to perform Hamiltonian simulation and Quantum Phase Estimation. The FFFT will be explained later on in this appendix.

E.1.2 Taylorization 'database' algorithm

Alternatively, we may use the Taylorization procedures from appendix B. Let us start with the 'database' algorithm. To carry it out we need to define how to perform the Prepare(W) and Select(H) operators.

Select(H) is virtually the same as the same preparation method as we describe in appendix D [5], except that in this case we use the notation of p odd or even for up and down spin values:

$$\text{Select}(H)\,|p,q,b\rangle\,|\psi\rangle = |p,q,b\rangle \otimes \begin{cases} Z_p\,|\psi\rangle & p = q \\ Z_p Z_q\,|\psi\rangle & (b=0)\wedge(p\neq q) \\ X_p \vec{Z} X_q\,|\psi\rangle & (b=1)\wedge(p>q)\wedge(p\oplus q=0) \\ Y_p \vec{Z} Y_q\,|\psi\rangle & (b=1)\wedge(p<q)\wedge(p\oplus q=0) \\ |\psi\rangle & (b=1)\wedge(p\oplus q=1) \end{cases}. \tag{70}$$

where $\oplus$ indicates sum modulus 2; and can therefore be implemented at cost $12N + 8\lceil\log_2 N\rceil + O(1)$ T gates.

Since the Prepare(W) method is not specified in the main reference for this appendix [6], we will also use the method from [5].

E.1.3 Taylorization 'on-the-fly' algorithm

In appendix K of [6] it is explained how to use the 'on-the-fly algorithm' in this context, which is similar to what we explained in appendix B [3].

The amplitudes we want to prepare, $W_{p,q,b}$, can be divided in a sum

$$W_{p,q,b} = \sum_{\nu\neq 0} W_{p,q,b,\nu}, \tag{71}$$

where

$$W_{p,q,b} = \begin{cases} \sum_{\nu\neq 0}\left(\frac{\pi}{2\Omega k_\nu^2}-\frac{k_\nu^2}{8N}+ \right. & p=q \\ \left.\frac{\pi}{\Omega}\sum_j \zeta_j \frac{\cos[k_\nu\cdot(R_j-r_p)]}{k_\nu^2}\right) & \\ \frac{\pi}{4\Omega}\sum_{\nu\neq 0}\frac{\cos[k_\nu\cdot r_{p-q}]}{k_\nu^2} & (b=0)\wedge(p\neq q) \\ \frac{1}{4N}\sum_\nu k_\nu^2\cos[k_\nu\cdot r_{p-q}] & (b=1)\wedge(p\oplus q=0) \\ 1 & (b=1)\wedge(p\oplus q=1). \end{cases} \tag{72}$$

If we have to sum over a large number of atoms J, we may also decompose each of the terms in the j sum independently.

Since it is easy to apply phases but not to change the amplitudes of a given state, [6] proposes further dividing each

$$W_{p,q,b,\nu} \approx \zeta \sum_{m=0}^{M-1} W_{p,q,b,\nu,m}; \quad W_{p,q,b,\nu,m}\in\{\pm 1\};$$
$$\zeta = \Theta\left(\frac{\epsilon}{\Gamma t}\right); \quad M\in\Theta\left(\frac{\max_{p,q,b,\nu}|W_{p,q,b,\nu}|}{\zeta}\right). \tag{73}$$

To perform the logic of the on-the-fly algorithm we first have to perform the calculations for the coefficients, which means we need costly arithmetic operations:

$$\text{Sample}(W)\,|p,q,b,\nu\rangle\,|0\rangle^{\otimes\lceil\log_2 N\rceil} \mapsto |p,q,b,\nu\rangle\,|\tilde{W}_{p,q,b,\nu}\rangle\,, \tag{74}$$

with $W_{p,q,b,\nu}$ a binary approximation to $\tilde{W}_{p,q,b,\nu}$.

The complexity will be $O(N^3+\log_2\epsilon_M^{-1})$, where the ϵ_M appears due to the use of Subprepare techniques from [5].

E.2 How to compute its cost

E.2.1 Trotterization algorithm

In this subsection we aim to explain the cost of performing Trotterization using this approach. To do so, we have to compute the cost of the FFFT operator, as well as the number of single qubit rotations in the exponentials and the number of segments required, r.

Let us start with the computation of the cost of FFFT. From [23] it can be seen that the number of gates required to perform an m-mode Fourier Transform are are $(m/2)\lceil\log_2(m/2)\rceil$ single qubit rotations and $(m/2)\lceil\log_2 m\rceil$ F_2 gates. The matrix representation of F_2 in the Jordan-Wigner representation is

$$F_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2^{-1/2} & 2^{-1/2} & 0 \\ 0 & 2^{-1/2} & -2^{-1/2} & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2^{-1/2} & 2^{-1/2} & 0 \\ 0 & 2^{-1/2} & -2^{-1/2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{75}$$

Therefore, we can see that F_2 is the product of a matrix that we will call W with a Control-Z. The gate W works as a Hadamard in the subspace spanned by $\{|01\rangle, |10\rangle\}$. Any gate with the structure of a unitary gate U in that subspace can be constructed as $C-U$ between two C-Nots in the opposite direction. In this case, U is the Hadamard gate, and the controlled-Hadamard gate can be performed using $R_y(\pi/4)$, a C-Not, and $R_y(-\pi/4)$. Therefore, in total F_2 requires two T gates in the Jordan-Wigner representation.

Overall, the FFFT requires $(N/2)\log_2(N/2) = (N/2)(\log_2 N - 1)$ single qubit z-rotations and $(N/2)\log_2(N)$ F_2 gates, as can be seen from figure 1b from [23].

The next step is computing the cost of the exponential rotations in (69). There are $8N$ terms in U, $8N(8N-1)/2$ terms in V and $8N$ terms in T in (68), so the same number of R_z rotations for operators T and U. Notice that in the simulation of $e^{-iV\tau}$ we will need Clifford gates and a single $C-R_z$ rotation per term [30, 51], as it was the case in appendix A.

Finally we want to compute the number of time segments in the Trotter decomposition r. Using the equations 5 and 6 from [55] we can see that the error in each time step is bounded by

$$2\left([T,[T,U+V]]+[(U+V),[T,(U+V)]]\right)\delta_t^3. \tag{76}$$

This, in turn, can be bounded [6] by

$$2(\max_\psi|\langle\psi|T|\psi\rangle|^2\cdot\max_\psi|\langle\psi|U+V|\psi\rangle|+ \max_\psi|\langle\psi|T|\psi\rangle|\cdot\max_\psi|\langle\psi|U+V|\psi\rangle|^2)\delta_t^3. \tag{77}$$

Since there are $r := t/\delta_t$ terms, the Trotter error is

$$\frac{\epsilon_{HS}}{r} \leq 2(\max_\psi|\langle\psi|T|\psi\rangle|^2\cdot\max_\psi|\langle\psi|U+V|\psi\rangle|+ \max_\psi|\langle\psi|T|\psi\rangle|\cdot\max_\psi|\langle\psi|U+V|\psi\rangle|^2)\left(\frac{t}{r}\right)^3. \tag{78}$$

Asymptotically, this means we will take

$$r = \Theta\left(\frac{\eta^2 N^{5/6}t^{3/2}}{\Omega^{5/6}\sqrt{\epsilon_{HS}}}\sqrt{1+\frac{\eta\Omega^{1/3}}{N^{1/3}}}\right). \tag{79}$$

We can find bounds for the expected values of U, V and T, in appendix F [6]. From equation F1

$$\max_{\psi} |\langle\psi|V|\psi\rangle| \leq \frac{2\pi\eta^2}{\Omega}\sum_{\nu\neq 0}\frac{1}{k_\nu^2} = \frac{\eta^2}{2\pi\Omega^{1/3}}\sum_{(\nu_x,\nu_y,\nu_z)\neq(0,0,0)}\frac{1}{\nu_x^2+\nu_y^2+\nu_z^2}, \tag{80a}$$

from F8

$$\max_{\psi} |\langle\psi|U|\psi\rangle| \leq \frac{4\pi\eta}{\Omega}\left(\sum_j \zeta_j\right)\sum_{\nu\neq 0}\frac{1}{k_\nu^2} = \frac{\eta^2}{\pi\Omega^{1/3}}\sum_{(\nu_x,\nu_y,\nu_z)\neq(0,0,0)}\frac{1}{\nu_x^2+\nu_y^2+\nu_z^2}, \tag{80b}$$

and from F10

$$\max_{\psi} |\langle\psi|T|\psi\rangle| \leq \frac{2\pi^2\eta}{\Omega^{2/3}}\nu_{\max}^2. \tag{80c}$$

To end up bounding U and V we need equation F6 [6]

$$\sum_{(\nu_x,\nu_y,\nu_z)\neq(0,0,0)}\frac{1}{\nu_x^2+\nu_y^2+\nu_z^2} \leq 4\pi\left(\sqrt{3}\frac{N^{1/3}}{2}-1\right) + \int_1^{N^{1/3}}\frac{3dz}{z^2}+\int_1^{N^{1/3}}\int_1^{N^{1/3}}\frac{3dxdy}{x^2+y^2} = 4\pi\left(\sqrt{3}\frac{N^{1/3}}{2}-1\right)+3-\frac{3}{N^{1/3}}+ \int_1^{N^{1/3}}\int_1^{N^{1/3}}\frac{3dxdy}{x^2+y^2}. \tag{81}$$

Using this and the previous equations, it is possible to calculate the actual value of r, given t and ϵ_{HS}.

E.2.2 Taylorization 'database' algorithm

Since the Prepare(W) method is not specified in the main reference for this appendix [6], we will also use the method from [5]. As explained in appendix D, the cost for Prepare(W) $6N+40\lceil\log_2 N\rceil+16\lceil\log_2\epsilon_{SS}^{-1}\rceil+10\mu$. Notice that the cost is linear because although there are $O(N^2)$ coefficients, only $O(N)$ are independent. In any case this will be multiplied by $\lambda = O(N^2)$.

Similarly taken from [5] and explained in appendix D the cost of Select(H) can be taken to be $12N + 8\lceil\log_2 N\rceil + O(1)$ T gates, since the implementation proposed in both references ([5] and [6]) is virtually the same.

E.2.3 Taylorization 'on-the-fly' algorithm

Finally, the main cost of the 'on-the-fly' algorithm comes from the Sample(W) operations that compute (72). This will require arithmetic operations as those indicated in table 2.

The main difference here will be calculating the value of λ', that influences the number of segments r. From equation K2 in [5] the Hamiltonian will have the form

$$H = \zeta\sum_{p,q,b,\nu,m} W_{p,q,b,\nu,m}H_{p,q,b} \tag{82}$$

Similarly as in previous appendices, we take

$$\zeta = \frac{\epsilon_H}{\Gamma r}. \tag{83}$$

In contrast to appendix B there is no integral over any volume, so we do not include $\mathcal{V}$ in the denominator; and in contrast to appendix C we do not sum over ρ so there is no division by μ. The main consequence of this form of preparing the initial state is changing the value of λ, that will now be, from eq. K5 in [6]

$$\lambda' = \zeta\sum_{p,q,b,\nu,m}|W_{p,q,b,\nu,m}|, \quad W_{p,q,b,\nu,m}\in\{-1,+1\}. \tag{84}$$

As a consequence, given that $m \in 0,...,M-1$, b can take values 0 and 1 and there are $8N$ values for p, q and ν

$$\lambda' = 2M\zeta(8N)^3. \tag{85}$$

Since

$$M = \frac{\max_{p,q,b,\nu}|W_{p,q,b,\nu}|}{\zeta} \tag{86}$$

we have that

$$\lambda' = 2(8N)^3\max_{p,q,b,\nu}|W_{p,q,b,\nu}| \tag{87}$$

As the sum of the nuclear charges is equal to the number of electrons $\sum_j\zeta_j = \eta$, we can bound $\max_{p,q,b,\nu}|W_{p,q,b,\nu}|$ as the maximum of 1 (the identity term);

$$\frac{\pi}{2\Omega k_\nu^2}-\frac{k_\nu^2}{8N}+\frac{\pi}{\Omega}\sum_j\zeta_j\frac{\cos[k_\nu\cdot(R_j-r_p)]}{k_\nu^2} \leq \frac{\pi}{2\Omega k_\nu^2}-\frac{k_\nu^2}{8N}+\frac{\pi\eta}{\Omega k_\nu^2} \leq \frac{\pi}{2\Omega k_\nu^2}+\frac{\pi\eta}{\Omega k_\nu^2}-\frac{k_\nu^2}{8N} = \frac{(2\eta+1)\pi}{2\Omega k_\nu^2}-\frac{k_\nu^2}{8N}; \tag{88a}$$

$$\frac{\pi}{4\Omega}\frac{\cos[k_\nu\cdot r_{p-q}]}{k_\nu^2} \leq \frac{\pi}{4\Omega k_\nu^2}; \tag{88b}$$

or

$$\frac{k_\nu^2}{4N}\cos[k_\nu\cdot r_{p-q}] \leq \frac{k_\nu^2}{4N}. \tag{88c}$$

Since the smallest value of $|k_\nu|$ for $\nu \neq 0$ is $k_\nu = 2\pi/\Omega^{1/3}$, and the largest is $k_\nu^2 = 3 \times \frac{(2\pi)^2 N^{2/3}}{\Omega^{2/3}}$

$$\max_{p,q,b,\nu} |W_{p,q,b,\nu}| \leq \max\left[\frac{(2\eta+1)}{8\pi\Omega^{1/3}} - \frac{\pi^2}{2N\Omega^{2/3}}, \frac{1}{8\pi\Omega^{1/3}}, \frac{6\pi^2}{N^{1/3}\Omega^{2/3}}\right], \tag{89}$$

Provided that the first option is the largest,

$$\lambda' \leq (8N)^3 \left(\frac{(2\eta+1)}{4\Omega^{1/3}\pi} - \frac{\pi^2}{N\Omega^{2/3}}\right). \tag{90}$$

Now we want to compute the number of arithmetic operations in the Prepare(w) operation. $p = q$ case of (72):

1. Calculating k_ν and r_p requires three multiplications each, one for each coordinate component, with $n = \lceil \log_2 N^{1/3} \rceil$.
2. There are three subtraction for each value of j in $R_j - r_p$ and another $r_{p-q} = r_p - r_q$, with $n = \lceil \log_2 N^{1/3} \rceil$.
3. Computing k_ν^2 requires 3 multiplications and 2 additions.
4. Calculating the product within the cosines costs three multiplications of length $n = \lceil \log_2 N^{1/3} \rceil$, and two sums between those terms.
5. One of the fastest ways to compute the cosine is to use the CORDIC algorithm [68], which requires a prefactor division (if expanded to a fixed order) and 2 sums per order since divisions by powers of two can be performed virtually.
6. We have to sum J cosine computations.
7. We have to divide or multiply such sum of cosines by a constant, and k_ν^2. Costs up to $\approx 3{\cdot}21 \log^2 N$.

Thus, the T-gate cost of this first calculation is $\approx J\left[\frac{35o}{2} + 63 + \frac{2o}{\log_2 N}\right] \log_2^2 N$, where J is the number of values of j, that indexes the atoms.

For $(b = 0) \wedge (p \neq q)$ and $(b = 1) \wedge (p \oplus q = 0)$:

1. We can reuse the previously calculated values of k_ν, k_ν^2 and compute r_q (3 multiplications) and r_{p-q} (3 subtractions).
2. We can perform the dot product in the cosine with 3 multiplications and 2 sums
3. Similarly, we have to perform a cosine calculation via the CORDIC algorithm again.
4. Finally we perform a multiplications and a division (by k_ν^2)

To perform the case $b = 1 \wedge (p+q = 0 \mod 2)$ we can reuse the cosine result from the previous point, as well as the k_ν^2 value, so we only need two multiplications.

E.3 How to adapt the Hamiltonian simulation to control the direction of the time evolution

E.3.1 Trotterization method

In the Phase Estimation protocol we should be controlling such rotations depending on the control ancilla qubits. However, since they are R_z rotations and $XR_z(\alpha)X = R_z(-\alpha)$ we can actually use a formulation similar to [5] where the mapping is $|1\rangle|\phi\rangle \to e^{i\phi}|1\rangle|\phi\rangle$ and $|0\rangle|\phi\rangle \to e^{-i\phi}|0\rangle|\phi\rangle$ (except for the first segment, but this is a minor cost). To control between both rotations we use C-Nots which change the direction of the Z rotation [74].

E.3.2 Taylorization methods

Adapting the Hamiltonian simulation for its use in Quantum Phase Estimation can be done as in appendix B.3. The cost can be therefore calculated in the same way.

F Trotter simulation: tighter bounds

In the previous appendix E we have explained how to perform Trotter simulation in plane waves. However, the bounds provided by (77) are somewhat loose, so the number of steps needed to achieve the same error are lower than required. Similarly happens for the methods covered in appendix A. In this appendix we give tighter bounds for the second order Hamiltonian simulation deterministic Trotter operator. We aim to approximate $e^{iH\delta_t}$, for $H = \sum_{\gamma=1}^{\Gamma} w_\gamma H_\gamma$, with

$$\mathcal{S}_2(H;\delta_t) = \left(\prod_{\gamma=1}^{\Gamma} e^{\frac{it}{2} w_\gamma H_\gamma}\right)\left(\prod_{\gamma=\Gamma}^{1} e^{\frac{it}{2} w_\gamma H_\gamma}\right). \tag{91}$$

This general expression will reduce, for the plane wave basis, to (69)

$$e^{-iH\delta_t} \approx e^{-i(U+V)\delta_t/2} \cdot FFFT^\dagger e^{-i(\delta_t/2)\sum_{\nu,\sigma} k_\nu^2 a_{\nu,\sigma}^\dagger a_{\nu,\sigma}} FFFT \cdot e^{-i(U+V)\delta_t/2} + O(\delta_t^3). \tag{92}$$

The error in this expression will be

$$\|e^{iH\delta_t} - \mathcal{S}_2(H;\delta_t)\| \leq W_2 \delta_t^3, \tag{93}$$

for $\delta_t = t/r$ and W_2 a commutator expression. Since in plane waves, operators U and V commute, in a Hamiltonian $H = T + U + V$ we only have to care about commutators $[[T, U+V], T]$ and $[[T, U+V], U+V]$. This can be better seen in the dual basis, where

$$V = \sum_{p \neq q} V_{pq} n_p n_q \tag{94}$$

and

$$U = \sum_p U_p n_p = \sum_p U_p n_p n_p, \tag{95}$$

for n_p the occupancy fermionic operator. Since the n_p operators commute with each other, so do U and V. Consequently, Ref. [63] proposes to write $H = T + \bar{V}$ with

$$\bar{V} := U + V = \sum_{p,q} \bar{V}_{p,q} n_p n_q. \tag{96}$$

One additional insight to bound the commutator W_2 as tightly as possible is to restrict our space to the space of η electrons. Usually, the error has been described in terms of the spectral norm distance, that is, in other words $\|H\|_2 = \max_\psi \| \langle\psi|H|\psi\rangle \|$. However, this takes into account states ψ that do not live in the subspace of η electrons, potentially leading to a higher norm and a looser bound. To remedy this, one can instead use the 'fermionic seminorm', defined as

$$\|H\|_\eta = \max_{\phi,\psi\in\mathcal{H}_\eta} \| \langle\phi|H|\psi\rangle \|, \tag{97}$$

for $\mathcal{H}_\eta$ the Hilbert subspace with η electrons. While this seminorm fulfills many properties of norms such as the triangle inequality, it is not a norm because some operators can evaluate to 0 without being operator 0 in the full Hilbert space, for example $\|n_p n_q\|_{\eta=1} = 0$.

Using the fermionic seminorm, we express the commutator error bound W_2 as [48]

$$W_2 \leq \frac{1}{12}\|[[T, U+V], T]\|_\eta + \|[[T, U+V], U+V]\|_\eta. \tag{98}$$

Furthermore, it is possible to bound each of the two terms independently as ([63] and appendix A in [48]):

$$\|[[T,\bar{V}],T]\|_\eta \leq 4\|T\|_2^2\|\bar{V}\|_{\max}\eta(4\eta+1) \tag{99}$$

$$\|[[T,\bar{V}],\bar{V}]\|_\eta \leq 12\|T\|_2\|\bar{V}\|_{\max}^2\eta^2(2\eta+1), \tag{100}$$

collectively known as the SHC bound, which scales as $O(N^3)$ with the number of basis functions N. Other bounds exist too (see sections 3 and 4, and table 1 in [48]), and shall be included in future updates to the library.

From equation 8 in [6], we also know that

$$\|U+V\|_{\max} \leq \frac{4\pi}{\Omega}\frac{\Omega^{2/3}}{4\pi^2}\sum_i \zeta_i = \frac{\Omega^{1/3}\eta}{\pi}, \tag{101}$$

while $\|T\|_2$ can be bounded as we did in (80c). From this, and the implementation cost of (69) that we discussed in appendix E.2, we can obtain an even lower cost of the Trotter simulation.

G Sparsity and low rank factorization

G.1 Method explanation

In the previous appendix we have seen that using carefully crafted Prepare and Select operators, it is possible to lower the complexity of the Quantum Phase Estimation. However, this came at the cost of having to use plane waves or similar basis sets. The method proposed in this appendix allows to leverage QROM techniques while working in arbitrary basis [5, 11]. The other main consideration of this article is how to leverage the sparsity and a low rank factorization of the Hamiltonian to lower the complexity of the algorithm.

Let us start by the second aspect, the low rank tensor factorization. We know that we can write the Hamiltonian in the second quantization in the following form

$$\begin{aligned} H &= \sum_{\sigma\in\{\uparrow,\downarrow\}}\sum_{p,q=1}^{N/2} h_{pq}a_p^\dagger a_q \\ &\quad + \frac{1}{2}\sum_{\alpha,\beta\in\{\uparrow,\downarrow\}}\sum_{p,q,r,s=1}^{N/2} h_{pqrs}a_{p,\alpha}^\dagger a_{q,\beta}^\dagger a_{r,\beta}a_{s,\alpha} \\ &= \sum_{\sigma\in\{\uparrow,\downarrow\}}\sum_{p,q=1}^{N/2} T_{pq}a_p^\dagger a_q \\ &\quad + \sum_{\alpha,\beta\in\{\uparrow,\downarrow\}}\sum_{p,q,r,s=1}^{N/2} V_{pqrs}a_{p,\alpha}^\dagger a_{q,\alpha}a_{r,\beta}^\dagger a_{s,\beta} \end{aligned} \tag{102}$$

The coefficients h_{pq} and h_{pqrs} are efficiently computable integrals. On the previous equation, the ordering $a^\dagger a^\dagger aa$ is called the 'physics notation' whereas the second ordering, $a^\dagger aa^\dagger a$ follows the chemists convention and will be the one we will use because it allows us to perform the factorization. Notice that T_{pq} and V_{pqrs} are real and have symmetries $p \leftrightarrow q$, $r \leftrightarrow s$ and $pq \leftrightarrow rs$. Notice also that the one-body operator changes as a result of the swapping of a_p and $a_p^\dagger$ and their anticommutation in the two-body term, and so does the sign of the latter.

Since V is a 4-rank tensor, with indices ranging from 0 to $N/2-1$, we can transform it to a $N^2/4 \times N^2/4$ matrix called W, with composite indices pq and rs, and symmetric and positive definite. Diagonalizing W we get,

$$Wg^{(l)} = w_l g^{(l)}; \quad W = \sum_{l=1}^{L} w_l g^{(l)}(g^{(l)})^T, \tag{103}$$

where $g^{(l)}$ denotes the l-th eigenvector, with eigenvalue w_l, and entries $g_{pq}^{(l)}$.

Let us denote the rank with L. If W where full rank, $L = N^2/4$. However, in most cases and due to Coulomb interaction being a two-body interaction, the rank will be $L = O(N)$. Now, we can rewrite

$$\begin{aligned} &\sum_{\alpha,\beta\in\{\uparrow,\downarrow\}}\sum_{p,q,r,s=1}^{N/2} V_{pqrs}a_{p,\alpha}^\dagger a_{q,\alpha}a_{r,\beta}^\dagger a_{s,\beta} \\ &= \sum_{l=1}^{L} w_l \left(\sum_{\sigma\in\{\uparrow,\downarrow\}}\sum_{p,q=1}^{N/2} g_{pq}^{(l)}a_{p,\sigma}^\dagger a_{q,\sigma}\right)^2. \end{aligned} \tag{104}$$

From the right-hand side of the equation we can see that there are $O(LN^2) = O(N^3)$ independent coefficients. In fact, due to the symmetry $p \leftrightarrow q$ there are $1/2 \cdot N/2(N/2-1)$ terms off diagonal, and when $p = q$ there are $N/2$ additional free coefficients. Therefore, in total there are $N^2/8 + N/4$ independent terms for each value of l. Further factorization is possible [39, 50, 69], but this work is not covered in this appendix.

As in the previous article, we do not attempt to perform phase estimation over $e^{\pm iH}$ but rather over $e^{\pm i \arccos(E_k/\lambda)}$, which is the phase produced by one step of the qubitization quantum walk. Also as in the previous article, this method uses Jordan-Wigner mapping too.

We have to explain how to perform operators Prepare and Select. Let us start with the former. The state we want to prepare is the following

$$|\psi\rangle = |0\rangle|+\rangle|0\rangle \sum_{p,q,\sigma} \sqrt{\frac{|T_{pq}|}{\lambda}} |\theta_{pq}^{(0)}\rangle |0\rangle |p,q,\sigma\rangle |0\rangle + \\ + \sum_l \sqrt{\frac{w_l}{\lambda}} |l\rangle |+\rangle |+\rangle \otimes \\ \otimes \sum_{p,q,r,s,\alpha,\beta} \sqrt{|g_{pq}^{(l)} g_{rs}^{(l)}|} |\theta_{pq}^{(l)}\rangle |\theta_{rs}^{(l)}\rangle |p,q,\alpha\rangle |r,s,\beta\rangle . \tag{105}$$

Here, $\theta_{pq}^{(l)}$ indicates the sign of each term, and are defined as

$$\theta_{pq}^{(l)} = \begin{cases} 0, & T_{pq} > 0, \\ 1, & T_{pq} < 0, \end{cases} \qquad \theta_{pq}^{(l)} = \begin{cases} 0, & g_{pq}^{(l)} > 0. \\ 1, & g_{pq}^{(l)} < 0. \end{cases} \tag{106}$$

We can see that the first register selects between the T terms (for state $|0\rangle$) and each of the L terms for $g^{(l)}$. The second and third register use $|+\rangle$ to select between $\mathbf{1}$, and $Z_{p,\sigma}$, $Z_{p,\alpha}$ and $Z_{q,\beta}$, whenever $p = q$ or $r = s$ respectively. Additionally, depending on whether $p > q$ or $p < q$ we apply $X_{p,\sigma}\vec{Z}X_{q,\sigma}$ or $Y_{p,\sigma}\vec{Z}Y_{q,\sigma}$ respectively.

The number of coefficients to fix is $(L+1)(N^2/8 + N/4)$, so the complexity will be $O(N^3 + \log_2 \epsilon_\mu^{-1})$, where the μ appears due to the use of Subprepare techniques from [5]. To perform the preparation, we follow this steps

1. Starting from the state $|0\rangle$, prepare a superposition over the first register

$$\left(|0\rangle \sqrt{\sum_{p,q} \frac{2|T_{pq}|}{\lambda}} + 2 \sum_l \sqrt{\frac{w_l}{\lambda}} |l\rangle \sum_{p,q} |g_{p,q}^{(l)}| \right) \otimes \\ \otimes |0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle . \tag{107}$$

   If we allow for error ϵ_{SS}, the complexity of this step, in terms of T-gates using the QROM is $4L + 4\mu + 14\lceil \log_2 L \rceil + 8\lceil \log_2 \epsilon_{SS}^{-1} \rceil$ [5]. The ϵ_{SS}^{-1} dependence is due to the Uniform operator preparation, that requires to use two controlled Z rotations, at cost $4\lceil \log_2 \epsilon_{SS}^{-1} \rceil$ each. On the other hand, the Uniform preparation requires $10\lceil \log_2 L \rceil$ T gates as can be seen from figure 12 in [5], which has to be added to $4\lceil \log_2 L \rceil$ T-gates due to the controlled-swap operations in Subprepare. The value of μ can be taken from equation 36 in [5].

2. Perform a Hadamard in the second register and another on the third, controlled on the first register being $|l > 0\rangle$.

$$\left(|0\rangle|+\rangle|0\rangle \sqrt{\sum_{p,q} \frac{2|T_{pq}|}{\lambda}} \right. \\ \left. + 2 \sum_l \sqrt{\frac{w_l}{\lambda}} |l\rangle|+\rangle|+\rangle \sum_{p,q} |g_{p,q}^{(l)}| \right) \otimes \\ \otimes |0\rangle|0\rangle|0\rangle|0\rangle . \tag{108}$$

   The cost of this step is negligible compared with the following one, and can be performed using a multicontrolled Hadamard.

3. Prepare a superposition over register six with amplitudes $\sqrt{|T_{pq}|}$ if $|l = 0\rangle$ or $\sqrt{|g_{pq}^{(l)}|}$ if $|l > 0\rangle$.

$$|0\rangle|+\rangle|0\rangle \sum_{p,q,\sigma} \sqrt{\frac{|T_{pq}|}{\lambda}} |0\rangle|0\rangle|p,q,\sigma\rangle|0\rangle + \\ + \sqrt{2} \sum_l \sqrt{\frac{w_l}{\lambda}} |l\rangle|+\rangle|+\rangle \otimes \\ \otimes \sum_{p,q,\alpha} \sqrt{|g_{p,q}^{(l)}|} \sqrt{\sum_{r,s} |g_{r,s}^{(l)}|} |0\rangle|0\rangle|p,q,\alpha\rangle|0\rangle . \tag{109}$$

   This step and the following have the largest complexities, since we need to use the unary iterator and Subprepare circuit of [5]. We have to iterate over L, p, and q, and that gives a Toffoli complexity of $(L+1)N^2/4 - 1$ plus the cost of the comparison and the controlled swaps from Subprepare.

4. For $|l > 0\rangle$, prepare weights $\sqrt{|g_{rs}^{(l)}|}$ in register 7.

$$|0\rangle|+\rangle|0\rangle \sum_{p,q,\sigma} \sqrt{\frac{|T_{pq}|}{\lambda}} |0\rangle|0\rangle|p,q,\sigma\rangle|0\rangle + \\ + \sum_l \sqrt{\frac{w_l}{\lambda}} |l\rangle|+\rangle|+\rangle \otimes \\ \otimes \sum_{p,q,r,s,\alpha,\beta} \sqrt{|g_{p,q}^{(l)} g_{r,s}^{(l)}|} |0\rangle|0\rangle|p,q,\alpha\rangle|r,s,\beta\rangle . \tag{110}$$

In this step the Toffoli complexity is also $LN^2/4$ plus the cost of the compare and controlled swaps.

5. Finally, use the QROM to output $|\theta^{(l)}_{pq}\rangle$ and $|\theta^{(l)}_{rs}\rangle$ in registers four and five.

$$|0\rangle|+\rangle|0\rangle\sum_{p,q,\sigma}\sqrt{\frac{|T_{pq}|}{\lambda}}|\theta^{(0)}_{pq}\rangle|0\rangle|p,q,\sigma\rangle|0\rangle + \\ +\sum_{l}\sqrt{\frac{w_l}{\lambda}}|l\rangle|+\rangle|+\rangle\otimes \\ \otimes\sum_{p,q,r,s,\alpha,\beta}\sqrt{|g^{(l)}_{pq}g^{(l)}_{rs}|}|\theta^{(l)}_{pq}\rangle|\theta^{(l)}_{rs}\rangle|p,q,\alpha\rangle|r,s,\beta\rangle\,. \tag{111}$$

To alleviate the cost of this procedure we follow three procedures:

1. Leverage the $p\leftrightarrow q$ symmetry in T_{pq} and $g^{(l)}_{pq}$, which divides the cost by half. This can be done preparing initially

$$\sqrt{2}\sum_{p>q}\sqrt{|g^{(l)}_{pq}|}|p,q,\alpha\rangle+\sum_{p}\sqrt{|g^{(l)}_{pp}|}|p,p,\alpha\rangle\,. \tag{112}$$

Then, one can use the second register, in state $|+\rangle$ to swap $|p\rangle$ and $|q\rangle$ when $p\neq q$ or to apply either $\mathbf{1}$ or $Z_{p,\sigma}$ when $p=q$. This means that in step 3 we will have to prepare $(L+1)(N^2/8+N/4)$ entries, and in step 4, $L(N^2/8+N/4)$.

2. We can also reduce the preparation cost in the QROM by performing the comparison between the probability $|\text{keep}_j\rangle$ and an ancilla in uniform superposition, at the same time for all $l\in(0,...,L)$. The controlled swap between the register $|j\rangle$ and $|\text{alt}_j\rangle$ can also be performed for all values of l simultaneously.

3. The dominant cost is outputting $(2L+1)(N^2/8+N/4)$ qubits using the QROM [5]. The outputs will have a size $M=\lceil\log_2 N^2\rceil+\lceil\log_2\epsilon^{-1}_{QPE}\rceil+O(1)$ where $\lceil\log_2 N^2\rceil$ is the size of $|\text{alt}\rangle$ and $\mu=\lceil\log_2\epsilon^{-1}_{QPE}\rceil+O(1)$ $|\text{keep}\rangle$, the size of the probability register. The key aspect of this third point is substituting the QROM of [5] by another from [46] which allows to trade some gate complexity by space complexity. We will call it QROAM. Calling also $d=(2L+1)(N^2/8+N/4)$ the number of entries we must look in the QROAM (including steps 3, 4 and 5), and $k=2^n$ an arbitrarily chosen power of 2. Then the complexity of computing the QROAM is $\lceil d/k_c\rceil+M(k_c-1)$ uncomputing it in Prepare† is $\lceil d/k_u\rceil+k_u$, where the k_c and k_u in compute and uncompute respectively can be different.

   As an aside, we can indicate that if we were to use dirty ancillae (anciallae that is already being used for other purposes) the cost would be $2\lceil d/k\rceil+4M(k-1)$ and $2\lceil d/k\rceil+4k$ for compute and uncompute respectively.

   Since the largest bottleneck is in the number of Toffolis required, we will focus on minimizing that variable. This means taking $k\approx\sqrt{d/M}$ for compute and $k\approx\sqrt{d}$ for the uncompute step, what means a cost of $2\sqrt{dM}$ and $2\sqrt{d}$ respectively, giving a total cost of $2\sqrt{d}(\sqrt{M}+1)$. Since we have chosen $d\approx LN^2/8$ and $M\approx\lceil\log_2(N^2)\rceil+\mu$, this means an overall cost $\sqrt{LN^2(\lceil\log_2(N^2)\rceil+\mu)/2}$ and half as many ancillae. Since $L=O(N)$, the number of Toffolis is $O(N^{3/2}\sqrt{\lceil\log_2 N\rceil+\mu})$.

A technical detail is that since the QROAM requires a continous output register, we will compute a single continous register for (l,p,q)

$$s'=l(N^2/8+N/4)+p(p+1)/2+q \tag{113}$$

The second operator we have to explain is Select, which is decomposed in two, Select$_1$ and Select$_2$ [11], performed again similarly as is done in appendix D [5]. The cost of this procedure is not dominant, as it will have complexity $O(N)$. From the representation of Select$_1$ in Figure 1 of [11], we can see that we need two QROM applications, as well as 2 equality comparisons.

Apart from the implementation of Prepare and Select, some other minor costs to have in mind are

- The cost in Select of each ranged operation is N, and each inequality test is $\lceil\log_2 N\rceil$. Since these operations have to be performed twice for (p,q) and again twice for (r,s), the total cost is $4N+4\lceil\log_2 N\rceil$.

- In the Prepare operator we have to initially prepare superpositions over $l\leq L$, $q\leq p<N/2$, $s\leq r<N/2$. We propose doing this by using the Uniform routine from the previous appendix (figure 12 in [5]). The initial uniform superposition over l requires $8\log_2 L+8\log_2\epsilon^{-1}_{SS}$ T gates. Enforcing an uniform superposition (in the Subprepare method) where $p\geq q$ requires a different method. We will slightly modify the suggestion of [11] to control the number of Amplitude Amplification steps. We do this by implementing the Uniform protocol both for p and q independently, and then adding an ancilla to check whether $p\geq q$. The success probability will be $\frac{N^2/8+N/4}{N^2/4}$ which approaches 1/2 from above. Since we cannot straightforwardly amplify that we add a second ancilla with success amplitude $\frac{1}{2}\sqrt{\frac{N^2/4}{N^2/8+N/4}}$. As a consequence, the product of success probabilities will be 1/2 that corresponds to a Grover's $\theta=\pi/6$ which can be amplified to amplitude 1 with a single step. So this step requires 2 Uniform$_{N/2}$ procedures and 1 ancilla rotation, to be performed thrice: preparation and

twice for Grover step. This second procedure has to be repeated twice to account for r and s too.

- The inequality test in state preparation has cost μ Toffolis due to the μ bits in $|\text{keep}\rangle$; and the same number of gates as qubits needed in the swap gate. We have to perform swap gates in the preparation procedure in the QROM where the register $|l,p,q\rangle$ has size $\lceil \log_2 L\rceil + 2\lceil \log_2(N/2)\rceil$. Then, we have to perform the same swap for $|r,s\rangle$ controlling on $l>0$, with registers of size $2\lceil \log_2(N/2)\rceil$. This means a Toffoli cost $\mu + \lceil \log_2 L\rceil + 4\lceil \log_2 N/2\rceil$. Here $\mu \approx \left\lceil \log\left(\frac{2\sqrt{2}\lambda}{\epsilon_{QPE}}\right)\right\rceil$.
- For state preparation remember that we only prepare those states that have $p>q$ and then use a controlled swap. These controlled swap for (p,q) and (r,s) cost $2\lceil \log_2 N/2\rceil$ Toffolis.
- The arithmetic operations for computing (113) require $2(\lceil \log_2 N/2\rceil)^2$ Toffolis gates.

In any case, the leading cost of the model is $\sqrt{LN^2(\log(N^2)+\mu)/2}$ Toffoli gates due to the QROAM. We can further reduce the cost by increasing the sparsity of the V operator, by zeroing all the terms $|V_{p,q,r,s}|<c$. Choosing c should be done in a way that does not affect the final error ΔE, as will be done choosing L too. To do that, the main aspect is substituting the QROM indexing

$$\frac{1}{\sqrt{d}}\sum_{j=1}^{d}|j\rangle\,|\text{alt}_j\rangle\,|\text{keep}_j\rangle \tag{114}$$

by another

$$\frac{1}{\sqrt{d}}\sum_{j=1}^{d}|j\rangle\,|\text{ind}_j\rangle\,|\text{alt}_j\rangle\,|\text{keep}_j\rangle \tag{115}$$

where ind_j indicates the j-th non-zero index, and d the number of non-zero terms in each case. This means that the swapping must now be performed between ind_j and alt_j. In this case we cannot simplify $\lceil d/k_c\rceil + (k_c-1)M + \lceil d/k_u\rceil + k_u$ with $M = \mu + 2\log_2 N + 2 \approx \log_2(N^2)+\mu$, directly to $\sqrt{LN^2(\log(N^2)+\mu)/2}$. The 2 in M is because we have to choose between T_{pq}, $g^{(l)}_{pq}$ and $g^{(l)}_{rs}$.

G.2 How to compute its cost

Notice that in contrast to other appendices, this calculation was already present in the original article [11] so the method to compute the cost is not our contribution. Only the automatisation of the computation is.

1. Steps 1 and 2 in state preparation can be performed using a QROM for L values and a multicontrolled-Hadamard gate respectively.
2. The largest cost in each step is the use of the QROAM for steps 3, 4 and 5, that as we saw is $\lceil d/k_c\rceil + M(k_c-1) - \lceil d/k_c\rceil + k_c$ Toffolis. It takes into account both the Prepare and Prepare† operators. We also use this step to prepare step 5, registers $|\theta_{pq}\rangle$ and $|\theta_{rs}\rangle$.
3. Here
   - $d = (2L+1)(N^2/8+N/4)$, as we take into account both steps 3 and 4 at the same time,
   - L is the rank of W. If W were full rank, $L = N^2/4$, but since W has a lot of structure $L = O(N)$.
   - $k_c \approx \sqrt{d/M}$ (closest power of 2),
   - $k_u \approx \sqrt{d}$ (closest power of 2),
   - $M = \log_2 N^2 + \mu$,
   - and $\mu \approx \left\lceil \log\left(\frac{2\sqrt{2}\lambda}{\epsilon_{QPE}}\right)\right\rceil$,
4. Each step requires to use Select once, at cost $4N + 4\lceil \log_2 N\rceil$.
5. At each Prepare we have to use Uniform three times: for l (accounted for in point 1 of this list), and two copies of $\tilde{s}$ for (p,q) and (r,s).
6. Other minor contributions of the Subprepare circuit (see [5]) include a μ-bit comparison and a $\log_2(LN^2/4)$-bit controlled swap.
7. The calculation of (113), which is carried out for pairs (p,q) and (r,s) can be done from the value of $\tilde{s}$ with 2 multiplications and three multiplications.
8. We need to perform amplitude amplification to prepare Uniform superposition over $p\geq q$ and $r\geq s$. This requires 6 Uniform$_{N/2}$ (for p and q and the three times of Amplitude Amplification), thrice an arbitrary rotation of the ancilla, thrice comparison between registers $|p\rangle$ and $|q\rangle$, and 2 Multi-controlled Z; and similarly for r and s respectively.

H Interaction picture

H.1 Method explanation

Although in previous appendices we have explored both the plane wave and Gaussian basis, there are two characteristics we have maintained constant over all the previous methods: all simulations were done in the Schrödinger picture and second quantization. These changes in later articles [7, 44, 62], and in this appendix, we present how the interaction picture can help make more efficient Hamiltonian Simulation algorithms [44].

Let us recall that the Schrödinger picture time evolution is dictated by the solution to the Schrödinger equation

$$\partial_t |\psi(t)\rangle = -iH(t)|\psi(t)\rangle \tag{116}$$

what implies that

$$|\psi(t)\rangle_S = e^{-iHt/\hbar}|\psi(0)\rangle, \tag{117}$$

whenever the Hamiltonian is time independent. We can see that in this case it is the state the one that evolves in time.

On the other hand we have the Heisenberg picture, where the dynamics are included in the operators. As such we have

$$\frac{d}{dt}A(t) = \frac{i}{\hbar}[H, A(t)] + \left(\frac{\partial A}{\partial t}\right)_H. \tag{118}$$

If the Hamiltonian is time independent this becomes

$$A(t)_H = e^{iHt/\hbar}A(0)e^{-iHt/\hbar}. \tag{119}$$

An intermediate option is to choose the interaction or Dirac picture, where both the state and the operators become time dependent. In this case we divide the Hamiltonian in two parts $H_S = H_{S,0}+H_{S,1}$, where $H_{S,1}$ carries the complexity and time dependence of the Hamiltonian. Then, the quantum state will evolve as

$$|\psi\rangle_I = e^{iH_{S,0}t/\hbar}|\psi(0)\rangle \tag{120}$$

and the operators will evolve as

$$A(t)_I = e^{iH_{S,0}t/\hbar}A(0)e^{-iH_{S,0}t/\hbar}. \tag{121}$$

In particular

$$H(t)_I = e^{iH_{S,0}t/\hbar}H_{S,1}e^{-iH_{S,0}t/\hbar}. \tag{122}$$

If the Hamiltonian is time-independent, we can evolve the state using e^{-iHt}, but if it is time-dependent, there is no closed expression in general. The time evolution operator is

$$U(t) = \lim_{r\to\infty}\prod_{j=1}^{r} e^{-iH(t(j-1)/r)\tau} := \mathcal{T}e^{-i\int_0^t H(s)ds}. \tag{123}$$

The authors of [44] explore two topics. In the first place, they build a time-dependent Hamiltonian simulation algorithm that is based on synthesizing a Dyson series. The second part of the article analyses how to apply the previous algorithm to simulate a Hamiltonian in the interaction picture. In particular, this allows us to simulate $e^{-i(H_{S,0}+H_{S,1})t}$ using

$$O(\lambda_1 t \text{poly}\log((\lambda_0+\lambda_1)t/\epsilon)) \tag{124}$$

queries to an oracle

$$(\langle 0|_a \otimes \mathbf{1}_s)O_1(|0\rangle_a \otimes \mathbf{1}_s) = \frac{H_{S,1}}{\lambda_1}, \tag{125}$$

and a similar amount of $e^{-iH_{S,0}\tau}$ queries with $\tau = O(\lambda_1^{-1})$. Here we were taking $\lambda_0 \geq ||H_{S,0}||$ and $\lambda_1 \geq ||H_{S,1}||$. If we had used the Schrödinger picture we would have instead needed

$$O((\lambda_0+\lambda_1)t\text{poly}\log((\lambda_0+\lambda_1)t/\epsilon_{HS})) \tag{126}$$

queries to oracles O_0 and O_1 of the form of (125). If $||H_{S,0}|| \gg ||H_{S,1}||$, and the complexity of applying $e^{-iH_{S,0}t}$ is similar to O_1, the interaction picture algorithm is advantageous.

Finally, the article applies the algorithm to the generalized Hubbard model and the electronic Hamiltonian, with a final complexity $\tilde{O}(N^2t)$.

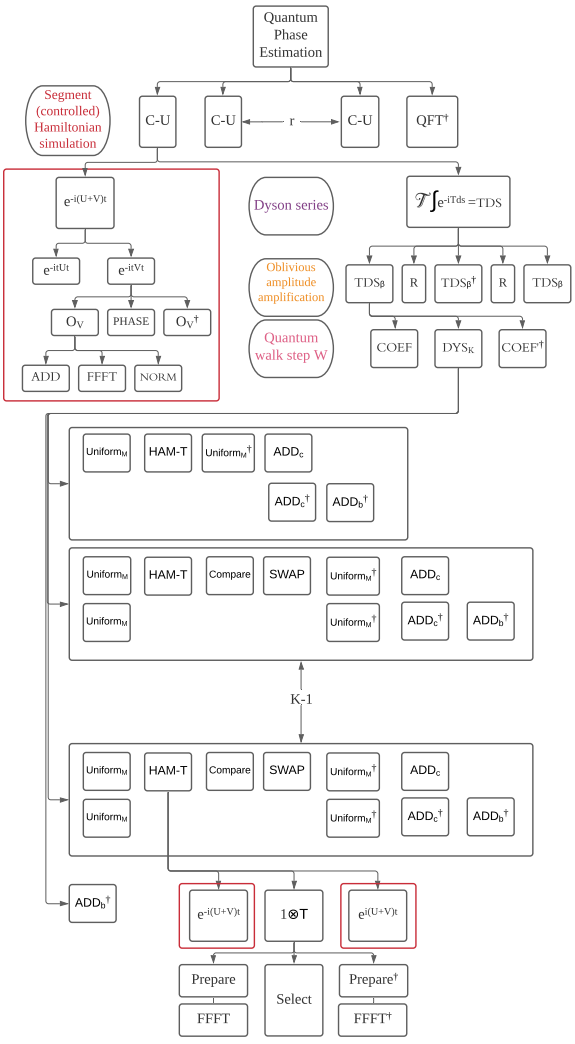

Figure 8: Abstraction level decomposition of the interaction picture protocol of [44]. The boxes in red represent the same protocol, only decomposed for one of them.

In contrast with previous algorithms, we cannot approximate $U(t)$ with a Taylor series unless $[H(t), H(t')] = 0$. The alternative is the Dyson series that converges absolutely whenever $t > 0$ and

bounded $||H(t)||$:

$$U(t) = \mathbf{1} - i\int_0^t H(t_1)dt_1 - \int_{t_2}^t\int_0^{t_2} H(t_2)H(t_1)dt_1dt_2 + i\int_{t_3}^t\int_{t_2}^{t_3}\int_0^{t_2} H(t_3)H(t_2)H(t_1)dt_1dt_2dt_3 + ... \tag{127}$$

We can rewrite the previous expression using the time ordering operator

$$U(t) = \mathcal{T}[e^{-i\int_0^t H(s)ds}] = \sum_{k=0}^{\infty}(-i)^k D_k. \qquad D_k = \frac{1}{k!}\int_0^t ...\int_0^t \mathcal{T}[H(t_k)...H(t_1)]d^kt. \tag{128}$$

As we did for the Taylor series, we have to truncate the series to order K such that the error remains lower than target ϵ_{HS}. We will see that K will be logarithmic in the corresponding precision.

Let us now focus on the input model. We need two definitions. The first is the usual block encoding

$$\text{HAM} = \begin{pmatrix} H/\lambda & \cdot \\ \cdot & \cdot \end{pmatrix} \Rightarrow (\langle 0|_a \otimes \mathbf{1}_s)\text{HAM}(|0\rangle_a \otimes \mathbf{1}_s) = \frac{H}{\lambda} \tag{129}$$

where we decompose HAM as in previous appendices

$$HAM = (\text{Prepare}^\dagger \otimes \mathbf{1}_s)\text{Select}(\text{Prepare} \otimes \mathbf{1}_s) \tag{130}$$

For a time dependent Hamiltonian we similarly define HAM-T as substituting H in the matrix form of HAM in (129) with

$$H = \text{Diagonal}[H(0), H(t/M), ..., H(1 - t/M)]. \tag{131}$$

In other words:

$$(\langle 0|_a \otimes \mathbf{1}_s)\text{HAM-T}(|0\rangle_a \otimes \mathbf{1}_s) = \sum_{l=0}^{M-1} |m\rangle\langle m| \otimes \frac{H\left(\frac{mt}{M}\right)}{\lambda}. \tag{132}$$

Having defined the main constructions for our algorithm, HAM and HAM-T, we now need the main theorem for simulating a time-dependent Hamiltonian for a short time segment:

Theorem 1. *[44] Let $H(s)$ be a time-dependent Hamiltonian such that $\max_s ||H(s)|| \leq \lambda$ and $\langle||\dot{H}||\rangle$ the average value of its time derivative. Let $M \in O\left(\frac{t^2}{\epsilon_{HS}}(\langle||\dot{H}||\rangle + \max_s ||H(s)||^2)\right)$. Then, for all $t \in [0, \frac{1}{2\lambda}]$ and $\epsilon_{HS} > 0$, exists W such that $||W - \mathcal{T}[e^{-i\int_0^t H(s)ds}]|| \leq \epsilon_{HS}$ with probability $1 - O(\epsilon_{HS})$, and $K = O\left(\frac{\log \epsilon_{HS}^{-1}}{\log\log \epsilon_{HS}^{-1}}\right)$ queries to HAM-T.*

The proof is given in Appendix B [44]. The key idea is that we want to approximate the time evolution operator with $W = \text{TDS}$, the oblivious amplitude amplification of $\text{TDS}_\beta = \sum_{k=0}^K \frac{(-it)^k}{M^k\beta} B_k$. As customary to require a single step of oblivious amplitude amplification, one takes $\beta = 2$.

The general strategy for TDS_β is similar to the Prepare Select Prepare† scheme. For the Select operator we first construct a sequence of K unitaries $U_1...U_K$ block-encoding matrices $H_1...H_K$:

$$(\langle 0|_a \otimes \mathbf{1}_s)U_k(|0\rangle_a \otimes \mathbf{1}_s) = H_k; \quad ||H_k|| \leq 1. \tag{133}$$

The consecutive applications of such matrices, $H_k...H_1 \propto B_k$, the k-th term in the Dyson series. DYS_K is the Select-like unitary that will apply this sequence $U_1...U_k$ controlled on index $|k\rangle$,

$$(\langle 0| \otimes \mathbf{1})DYS_K(|0\rangle \otimes \mathbf{1}) = \sum_{k=0}^K |k\rangle\langle k| \otimes \gamma_k B_K, \tag{134}$$

where $\gamma_k = M^{-k}$ will be a weighting coefficient of the Dyson series. Constructing such U_k operators is explained in the appendix B [44].

The second ingredient needed is Prepare-like operators $COEF$ and $COEF'^\dagger$, the difference between them being the phase in the Dyson series term. This allows us to perform the TDS_β operator (see fig. 6). Uniform_M, needed in the implementation of DYS_K can be implemented as suggested in the main reference for this appendix [5], while the rest are arithmetic operations, and the implementation of HAM-T discussed later on.

To extend Theorem 1 to longer time periods one can just apply it multiple times with the corresponding scaled error, as given by Corollary 4 of [44]. Since Theorem 1 indicates that the maximum simulation time for a single segment is $\tau = t/\lceil 2\lambda t\rceil$ with $\max_s ||H(s)|| \leq \lambda$, the number of segments is $r = \lceil 2\lambda t\rceil$, and the error allowed for each segment $\delta = \epsilon_{HS}/r$. Then Lemma 5 in [44] states that the constants K and M for the simulation of a single segment to error δ are

$$K = \left\lceil -1 + \frac{2\log(2r/\epsilon_{HS})}{\log\log(2r/\epsilon_{HS}) + 1}\right\rceil \tag{135a}$$

and

$$M = \left\{\frac{16\tau^2}{\delta}(\langle||\dot{H}||\rangle + \max_s ||H(s)||^2), K^2\right\}. \tag{135b}$$

The next step is to use this framework to simulate time-independent Hamiltonians in the interaction picture

$$H_I(t) = e^{iH_{S,0}t}H_{S,1}e^{-iH_{S,0}t}. \tag{136}$$

The advantage of simulating in this frame will happen when the norm of $||H_I(t)|| = ||H_{S,1}|| \ll ||H_{S,1}|| + ||H_{S,0}||$. We can apply this formalism to the Hamiltonian in the dual wave basis, equation (68), where $H_{S,0} = U + V$ and $H_{S,1} = T$.

To simulate our time-independent Hamiltonian we use the Theorem 1 and Corollary 4 in [44]. As

$$\begin{aligned}|\psi_S(t)\rangle &= e^{-iH_{S,0}t}\,|\psi_I(t)\rangle \\ &= e^{-iH_{S,0}t}\mathcal{T}[e^{-i\int_0^t H_I(s)ds}]\,|\psi(0)\rangle\end{aligned} \tag{137}$$

we can divide the evolution in r segments, $\tau = t/r$,

$$\begin{aligned}|\psi(t)\rangle &= (e^{-i(H_{S,0}+H_{S,1})\tau})^r\,|\psi(0)\rangle \\ &= \left(e^{-iH_{S,0}\tau}\mathcal{T}[e^{-i\int_0^\tau H_I(s)ds}]\right)^r|\psi(0)\rangle\,,\end{aligned} \tag{138}$$

and simulate it in a similar way as suggested by Corollary 4 in [44]. We have already explained ow to perform $\mathcal{T}[e^{-i\int_0^\tau H_I(s)ds}]$. However, we have yet to specify how implement HAM-T, which can be done as

$$\begin{aligned}\text{HAM-T} = &\left(\sum_{m=0}^{M-1}|m\rangle\langle m|_d\otimes\mathbf{1}_a\otimes e^{iH_{S,0}\tau m/M}\right)\cdot \\ \cdot(\mathbf{1}_d\otimes H_{S,1})\cdot&\left(\sum_{m=0}^{M-1}|m\rangle\langle m|_d\otimes\mathbf{1}_a\otimes e^{-iH_{S,0}\tau m/M}\right).\end{aligned} \tag{139}$$

We also have to explain how to implement $e^{-i(U+V)t}$. Notice that U and V commute because they are diagonal in the dual wave basis and can therefore be fast-forwarded. However, to avoid the $O(N^2)$ cost in the V term we will

1. Define the Fourier transform of V coefficients, $\tilde{V}(\vec{k}) = \sum_{\vec{x}} V(\vec{x})e^{2\pi i\vec{x}\cdot\vec{k}/N^{1/d}}$, and of the operators, $\tilde{\chi}_{\vec{k}} = \frac{1}{\sqrt{N}}\sum_{\vec{x}} e^{-2\pi i\vec{x}\cdot\vec{k}/N}\sum_\sigma n_{\vec{x},\sigma}$; and assuming V is real and symmetric, equation 39 from [44] writes

$$\begin{aligned}V &= \sum_{(\vec{x},\sigma)\neq(\vec{y},\sigma')} V(\vec{x}-\vec{y})n_{\vec{x},\sigma}n_{\vec{y},\sigma'} \\ &= \sum_{\vec{k}}\tilde{V}(\vec{k})\tilde{\chi}_{\vec{k}}\tilde{\chi}_{\vec{k}}^\dagger + \sum_{\vec{p},\sigma}\left(\sum_{\vec{k}}\tilde{V}(\vec{k})\right)n_{\vec{p},\sigma}.\end{aligned} \tag{140}$$

2. Use a binary oracle O_A such that $O_A\,|j\rangle\,|0\rangle_o\,|0\rangle_{garb} = |j\rangle\,|A_j\rangle_o\,|g(j)\rangle_{garb}$. Then we can implement the phase operator

$$\begin{aligned}&|j\rangle\,|0\rangle_o\,|0\rangle_{garb} \to_{\tilde{O}_A} |j\rangle\,|A_j\rangle_o\,|g(j)\rangle_{garb}\,|0\rangle \to_{PHASE} \\ &e^{-iA_jt}\,|j\rangle\,|A_j\rangle_o\,|g(j)\rangle_{garb}\,|0\rangle \to_{\tilde{O}_A^\dagger} e^{-iA_jt}\,|j\rangle\,|0\rangle_o\,|0\rangle_{garb}\end{aligned} \tag{141}$$

We want to implement the oracle O_V to calculate the Fourier Transform of V (omitting garbage registers), so this oracle can be decomposed as

$$\begin{aligned}\left(\bigotimes_{x,\sigma}|n_{x,\sigma}\rangle\right)|0\rangle &\to_{ADD}\bigotimes_x|\sum_\sigma n_{x,\sigma}\rangle \to_{FFT}\bigotimes_k|\tilde{\chi}_k\rangle \\ &\to_{|\cdot|^2}\bigotimes_k||\tilde{\chi}_k|^2\rangle \to_{\times V_k}\bigotimes_k|V_k|\tilde{\chi}_k|^2\rangle\,.\end{aligned} \tag{142}$$

Notice that $|n_x\rangle$ indicates the occupancy of the corresponding orbital, and as we are working with fermions, the FFT is the Fermionic Fast Fourier Transform.

H.2 How to compute its cost

To be able to count the complexity of the circuit it is useful to first indicate the size of each of the registers that appear in the algorithm, and more in particular in the analysis of the TDS operator in appendix B [44]:

1. Register s is the register containing the state. In second quantization it has size N.
2. Register a has a size given by the block encoding. Using [45] this can be bound by the logarithm of the number of unitary terms in Hamiltonian that are summed, $n_a = \lceil\log_2\Gamma\rceil$.
3. Register b has $n_b = \log_2(K+1)$ qubits.
4. Register c has $n_c = 1+\log_2(K+1)$ qubits.
5. Registers d and e require $\log_2 M$ qubits.
6. Register f only requires 1 qubit.

Secondly, we have to specify the value of K and M in (135). For K we already mentioned that $\delta = \epsilon_{HS}/r$, while in the usual definition of r we will take $\lambda = \lambda_1 = ||H_{S,1}|| = ||T||$. Instead of taking $\tau = \frac{1}{2\lambda_1}$ [44], we may take it slightly higher, $\tau = \frac{\ln 2}{\lambda_1}$ as this limit comes from the oblivious amplitude amplification technique [9], and we will do so to carry out similar treatment between the algorithms. Then, since $t = \frac{\pi}{\epsilon_{QPE}}$ this implies that $r := t/\tau = \lceil||T||t\rceil = \left\lceil\frac{\pi||T||}{\epsilon_{QPE}\ln 2}\right\rceil$.

Additionally, we need to obtain the value of M. Since $\max_s||H_I(s)|| \leq ||H_{S,1}||$ and $\langle||\dot{H}||\rangle = |||[H_{S,0},H_{S,1}]|| \leq 2||H_{S,0}||\cdot||H_{S,1}||$, substituting $\tau = \ln 2/\lambda_1$ and $\delta = \epsilon_{HS}/r = \epsilon_{HS}t/\tau$ in the value of M

$$\begin{aligned}M &= \max\left\{\frac{16\tau^2}{\delta}(\langle||\dot{H}||\rangle + \max_s||H(s)||^2), K^2\right\} \\ &= \max\left\{\frac{16t\ln 2}{\lambda_1\epsilon_{HS}}(2||H_{S,1}||||H_{S,0}|| + ||H_{S,1}||^2), K^2\right\} \\ &= \max\left\{\frac{16t\ln 2}{\epsilon_{HS}}(2||H_{S,0}|| + ||H_{S,1}||), K^2\right\}.\end{aligned} \tag{143}$$

To finish giving a description of the algorithm we need to particularize HAM-T for the time-independent Hamiltonian that we want to use, as

given in Lemma 7 in [44]

$$\text{HAM-T} = \left(\sum_{m=0}^{M-1} |m\rangle\langle m| \otimes \mathbf{1}_a \otimes e^{i(U+V)\tau m/M}\right) \cdot \\ \cdot (\mathbf{1} \otimes O_T) \cdot \left(\sum_{m=0}^{M-1} |m\rangle\langle m| \otimes \mathbf{1}_a \otimes e^{-i(U+V)\tau m/M}\right). \tag{144}$$

Here,

$$\left(\sum_{m=0}^{M-1} |m\rangle\langle m| \otimes \mathbf{1}_a \otimes e^{i(U+V)\tau m/M}\right)$$

can be implemented with $\lceil \log_2 M \rceil$ controlled-rotations of the kind $e^{i(U+V)\tau/M}$, $e^{i(U+V)\tau 2/M}$, $e^{i(U+V)\tau 4/M}$...

Lastly, performing O_T can be done using $O_T = (\text{Prepare}_T^\dagger \otimes \text{FFFT}^\dagger)\text{Select}_T(\text{Prepare}_T \otimes \text{FFFT})$, where

$$\text{Prepare}_T |0_a\rangle = \sum_p \sqrt{\frac{\tilde{T}(p)}{\lambda_T}} |\vec{p}\rangle, \tag{145a}$$

$$\text{Select}_T = \sum_p |\vec{p}\rangle\langle\vec{p}| \otimes n_p, \tag{145b}$$

and FFFT applied using $O(N \log N)$ gates, as we already discussed in appendix E.

Finally, let us highlight that there is a way to avoid the extra cost posed by Amplitude Amplification. The key idea is to implement a block encoding of $\sin(H\tau) = \frac{e^{+iH\tau} - e^{-iH\tau}}{2i}$, so that the qubitization walk operator will implement $e^{-i \arcsin \mu_j/\lambda'} = e^{-i \arcsin(\sin E_j\tau)/\exp(\lambda_1\tau)} \approx e^{-iE_j\tau/\exp(\lambda_1\tau)}$ [62], for $\mu_j = \sin E_j\tau$ and $\lambda' = \exp(\lambda_1\tau)$ [34]. If the segment time length of the amplitude amplified algorithm is $\tau \approx \ln 2/\lambda_1$, then the adjusted segment length is $\tau_{\text{eff}} \approx \ln 2/2\lambda_1$, and if one increases $\tau \approx 1/\lambda_1$, then $\tau_{\text{eff}} \approx 1/(e\lambda_1)$. This means that each step of the algorithm does not need to be amplified, but the number of time segments increases from $\lambda_1/(\epsilon_{QPE} \ln 2)$ to $e\lambda_1/\epsilon_{QPE}$ [62]. In this case we also aim to perform Hamiltonian simulation over $H - \tilde{E}_0$, where $\tilde{E}_0$ is an approximation to the ground state energy, so that we operate on the linear regime of the sine and arcsine functions, and the error introduced is small.

H.3 How to adapt the Hamiltonian simulation to control the direction of the time evolution.

We will use the trick of setting the controlled evolution of phase estimation $|1\rangle|\phi\rangle \to e^{i\phi}|1\rangle|\phi\rangle$ and $|0\rangle|\phi\rangle \to e^{-i\phi}|0\rangle|\phi\rangle$. This will be reflected in the Dyson expansion, where we will have to add a i factor to the coefficients in $COEF$ conditional on the phase estimation ancilla being on state 1; and in the sign of the exponential $e^{-i(U+V)\tau}$.

Part II
Quantum Chemistry

8

TFermion

One of the processes in quantum chemistry that can have the greatest impact is to estimate the ground state energy of a quantum system (GSEE). The ground state is the lowest possible energy state in a physical system, according to quantum mechanics. This fundamental energy is of particular importance because it affects the overall behavior of the system and can be crucial to understanding its properties.

To solve the GSEE problem, there are some methods to efficiently estimate the energy of a state. However, the problem arises because of the difficulty in preparing the ground state (GSP) of a quantum system. The GSP problem has been classically studied and some algorithms exist for it. However, these algorithms do not efficiently represent the information contained in a quantum system and, therefore, performing the evolution operations is so costly that simplifications far removed from reality are made. These simplifications are not able to produce reliable results, making both GSP and GSEE classically very limited.

Quantum computing has two fundamental advantages in quantum chemical algorithms. First, it allows a direct and realistic representation of quantum states without approximations. Second, a QC algorithm inherently simulates evolution. These two advantages are automatic because of the nature of quantum algorithms but, in addition, there may be other advantages related to state space exploration or data set processing. These advantages convert quantum chemistry in a solid candidate for quantum advantage. These advantages position quantum chemistry as a strong candidate for achieving quantum advantage. Nonetheless, it is crucial to investigate the quantum gate requirements of quantum chemistry algorithms to assess the current capability of quantum devices to execute them. For that purpose, TFermion library is presented.

8.1 Introduction to GSP and GSEE

One of the fundamental purposes of GSEE is to understand and predict the behavior of quantum systems, such as molecules in quantum chemistry or materials in condensed matter physics. The ability to estimate the ground state energy is essential for practical problems, such as the simulation of molecules for the design of new drugs or materials, where understanding the ground state is crucial to understanding their properties and behaviors.

As seen in other sections, GSP and GSEE algorithms can combine classical and quantum modules to create solutions that are executable in the near term on the quantum devices at hand. However, while this can be done, it is not a mainstream development in the scientific community. This is because the advantage of quantum algorithms for GSEE is very large. The effort is more focused on getting algorithms that require fewer and fewer resources than on combining them with other classical ones. In fact, if the concept of hybrid algorithm exists, it is simply because the objective is to find a quantum advantage of this type of algorithms as soon as possible. An example of hybrid quantum chemistry algorithms that will be discussed later is the Variational Quantum Eigensolver (VQE) that served as the basis for the variational algorithms explained in chapter 5.

In this thesis, a different approach is applied to GSEE quantum algorithms. This approach is not intended to result in a new algorithm, neither quantum nor hybrid. The goal of this new approach is to accelerate the development process of quantum algorithms and to better understand their performance.

In this work, a classical computing software library has been developed that gives information about the cost of several GSEE quantum algorithms to compare them with each other and to understand which techniques are better. Basides, to give information about the needs of quantum hardware to be able to execute these algorithms. All this has resulted in the creation of a software tool for measuring the performance of GSEE quantum algorithms called TFermion.

To explain how TFermion works and to show a practical example of its usefulness, as a use case, it is necessary to understand some intermediate concepts. First, the most important classical algorithms for GSP and GSEE will be briefly explained. Then, it is necessary to explain some Hamiltonian simulation techniques, which is the key process to represent the molecule and various forms of representation needed. Finally, some quantum algorithms for GSEE based on the representations explained above will be reviewed.

As discussed above, classical algorithms for GSP and GSEE make sense as long as there are no quantum algorithms that can be run on real devices. Still, some of these classical algorithms and their approximations have served as a fundamental basis for other quantum algorithms. The main limitations of classical quantum chemistry algorithms are found in the representation

of states, especially when they need to describe in great detail the behavior of some particles with quantum effects. Of this type, three theoretical descriptions stand out: Hartree-Fock (HF), density functional theory (DFT), and coupled cluster (CC).

The Hartree-Fock (HF) method is an approximate form of the quantum mechanical equations for fermions. HF method uses equations based on single-particle orbitals that are more computationally accessible than methods based on many-particle wave functions. It takes into account the interaction between electrons and the influence of their electromagnetic fields.

The HF method receives as input the electronic Hamiltonian and returns the molecular orbitals and their energies. It is an iterative method until convergence of the orbital values. It is important to note that the Hartree-Fock method is an approximation, it does not take into account the complete electronic correlation. Although it is an approximation, it has been useful as a basis for developing more sophisticated methods to describe molecular systems with greater accuracy.

Density Functional Theory (DFT) [121, 122] is a theoretical framework in quantum chemistry used to describe the electronic structure of atomic and molecular systems. Unlike the HF method seen above, it does not deal directly with electron wave functions but focuses on electron density as the fundamental quantity.

In DFT, the electron density represents the probability of finding an electron at a particular point in space. Upon that, DFT tries to minimize the total energy of the system by varying the electron density. DFT is based on the Kohn-Sham equation, which decomposes the system into non-interacting dummy electrons, called Kohn-Sham electrons, moving in an effective potential. The main limitation of DFT is that due to the computational complexity of calculating the quantum effects between the electrons, approximations are required that cause the system to lose accuracy.

DFT has proven to be a powerful and efficient tool for describing a variety of molecular and solid systems. It is widely used in theoretical chemistry and materials simulation, and has contributed significantly to the understanding of electronic structures and chemical properties.

The basic idea of the Coupled Cluster procedure is to expand the wave function as excitations of the HF state, which is an approximation to the fundamental state of a quantum system. The method employs cluster excitation operators that act on the reference state to create electronic excitations. These cluster operators are decomposed in terms of single-electron excitations (singles), double excitations (doubles) and so on. The cluster wave function is expanded exponentially using the cluster operator. This means that higher order terms take into account the interactions between multiple electrons.

CC is known for its high accuracy, especially in systems where electronic correlation is crucial.

However, it is also computationally intensive and its practical use is limited to systems of moderate size due to the complexity of the calculations. This method is also the basis for the Unitary Coupled-Cluster ansatz used for the Variational Quantum Eigensolver.

8.2 Hamiltonian simulation, representation, mapping and encoding for GSP and GSEE

In order to design quantum algorithms to simulate the evolution of the quantum chemistry system to be evaluated, a definition of that evolution is necessary. One such way of defining evolution is Hamiltonian simulation. This type of simulation gives the algorithm the ability to simulate the time evolution of a system described by a given Hamiltonian. This simulation makes it possible to study how the quantum system evolves in time.

By describing a quantum system with its Halmiltonian and being able to use it to describe a behavior, a more detailed study of its properties is performed. This results in an advantage in accuracy. In addition, quantum computers allow Hamiltonian simulation in a natural way, so this is another aspect in which a quantum advantage is expected to be obtained in a practical problem.

The Trotter-Suzuki decomposition was the first method proposed to implement a Hamiltonian simulation. It is based on the decomposition of the time evolution operator into a series of more manageable operators. The main idea is to divide the time into small steps and express the time evolution as a product of simpler operators.

In this way, it is possible to apply the simple operators separately and then combine the result. This algorithm tries to make a trade-off between accuracy and performance. In such a way that higher precision implies lower performance, higher complexity of execution. The Trotter-Suzuki algorithm can be generalized to include higher order terms, improving the accuracy of the approximation. However, as the order increases, the computational complexity also increases.

Trotter's method is the most widespread and widely used method for Hamiltonian simulation. However, it has some points that can be improved such as the scalability of its error. For this purpose, methods known as post-Trotter have emerged. One of them is the Taylor series decomposition for Hamiltonian simulation.

The terms of the Taylor series can be realized by introducing an additional superposition and conducting controlled operations. The temporal evolution is divided into segments, each sufficiently short to allow for an accurate approximation of the evolution using a certain number of terms in the Taylor series. Each segment is then performed using oblivious amplitude amplification [66].

Another post-Trotter method is qubitization. Qubitization refers to the expression of the time evolution operator in terms of qubits. The key idea behind this algorithm is to find an efficient way to express the time evolution operator in local terms of quantum operations that can be efficiently implemented in a quantum circuit.

An example of representation, in the context of molecule simulation, is that the Hamiltonian terms can represent interactions between atoms or between electrons and nuclei, and the qubits represent the occupation of an orbital in a molecular simulation. Once the quantum system has been mapped to qubit space, other algorithms such as the Trotter-Suzuki algorithm can be applied. The main advantage of qubitization is that it is an errorless Hamiltonian simulation, but it is costly for a quantum computer.

In order to reduce the complexity of the implementation of the qubitization method, the coupled cluster method was developed. In this method, the time evolution is divided into two parts: the free part and the interaction part.

The free part describes the time evolution of the system without any interaction, using an unperturbed Hamiltonian. The interaction part describes how the system interacts, and its time evolution operator is constructed by combining the free part and the interaction part. This approach facilitates the treatment of complex systems by dividing the problem into more manageable parts.

In order to run the GSP and GSEE quantum algorithms, it is necessary to use the Hamiltonian describing the evolution, as seen above. In addition, it is necessary to represent the problem, with first or second quantization, to make a mapping between operators and qubits, by means of Jordan-Wigner or Bravyi-Kitaev, and to make a choice of a set of basis functions to generate the wave function, which can be gaussian functions or plane waves.

The representation of the states in a quantum circuit is one of the key decisions when implementing an algorithm. Classically, it is already an important decision because it largely determines whether the algorithm is valid or not; quantumly, it becomes even more important because of the need to optimize the number of qubits and operations. In this case, it should be discussed how to use the logical qubits, noise-free and fully connected to each other, to represent the possible quantum states of the system.

The first option is to use the first quantization. This representation indicates the state of each electron, i.e., for each electron it indicates in which orbital it is located. It is a compact representation because it only requires a number of qubits proportional to the number of electrons and the logarithm in base 2 of the number of orbitals. However, operating in this quantization introduces a certain computational overhead, which slows down the calculation.

The second option is to use the second quantization. This represents the occupation of each

orbital, i.e., in each orbital it is indicated how many electrons there are. This representation requires a number proportional to the number of orbitals in the system. If the number of orbitals is much larger than the number of electrons, the second quantization is much less efficient than the first. However, this representation does not require any additional calculations, so it is more efficient in performing operations.

To simulate the quantum system, once the representation has been chosen, it is necessary to map the system to the operators. In other words, designate a set of qubit operators (matrices) which satisfy the canonical anticommutation relations. In order to map fermions to qubits, and depending on the chosen quantization, there are two main methods to perform the mapping, Jordan-Wigner or Bravyi-Kitaev.

The Jordan-Wigner mapping is a theoretical method that uses second quantization to transform spin operators into fermionic creation and annihilation operators. This operation transforms "up" spins into fermions or occupied states, and "down" spins into unoccupied states. The cost of Jordan-Wigner mapping and parity mapping is $\mathcal{O}(N)$, where N is the number of qubits [123].

Alternatively, the Bravyi-Kitaev method ensures that the mapping can be done at a complexity equivalent to the logarithm in base 2 of the number of qubits. Balances the locality of the occupancy and parity information to improve the efficiency of the simulation. The qubit stores the parity of the set of occupancy numbers corresponding to that set of orbitals [124].

Once the representation and mapping methods were explained, it is necessary to define the set of basis functions to generate the wave function. The basis set can either be composed of atomic orbitals (yielding the linear combination of atomic orbitals approach), which is the usual choice within the quantum chemistry community; plane waves, which are typically used within the solid-state community; or real-space approaches. Several types of atomic orbitals can be used: Gaussian-type orbitals are by far the most often used, as they allow efficient implementations of post-Hartree–Fock methods.

In molecular calculations, a common practice is to utilize a basis comprising atomic orbitals centered at each nucleus within the molecule (linear combination of atomic orbitals ansatz). The most physically motivated basis sets are Slater-type orbitals (STOs), which are solutions to the Schrödinger equation for hydrogen-like atoms. However, computing integrals with STOs poses computational challenges, leading to the approximation of STOs as linear combinations of Gaussian-type orbitals (GTOs). Numerous Gaussian-type orbital basis sets have been documented in the literature [125], often organized in hierarchies of increasing size to provide a controlled approach for obtaining more accurate solutions, albeit at a higher computational cost.

The alternative to Gaussian basis functions is the use of plane waves, which are better suited for periodic materials because of their periodicity. Generally, the selection of the plane wave basis set is based on a cutoff energy. Plane waves within the simulation cell that satisfy the energy criterion are incorporated into the calculation. Moreover, certain integrals and operations are more straightforward to program and execute with plane-wave basis functions compared to their localized counterparts. The number of plane waves is typically around 100 times larger than the Gaussian functions, as recommended in Appendix E of [126].

Now that all the elements necessary to run a quantum algorithm based on the Quantum Phase Estimation (QPE) technique for GSP and GSEE have been explained, some of these algorithms will be briefly mentioned. Alternatively, the Variational Quantum Eigensolver (VQE) algorithm will also be reviewed because it is an alternative, although less efficient, method to the QPE-based algorithms.

The concept of variational algorithms for QML problems arises from the creation of a variational algorithm applied to quantum chemistry, the VQE. The Variational Quantum Eigensolver (VQE) is built upon the concept that it is possible ton prepare the ground state by variationally determining a state that minimizes the ground state energy.

In the variational algorithms scheme, a quantum ansatz and a classical optimizer are needed. In VQE, the classical optimizer is still a gradient descent, and the original quantum ansatz was Unitary Coupled-Cluster. Later, other ansatz derived from the original such as Qubit Coupled-Cluster was chosen. In VQE, as in all variational algorithms, the selection of the ansatz is key for the algorithm to find a solution close to the optimum. Otherwise, the barren plateaus phenomenon appears and taking a step requires an exponential number of intermediate operations.

The Quantum Phase Estimation (QPE) technique allows to know the ground state of a quantum system and its energy with greater precision than other classical and quantum techniques. However, it is very expensive to implement. Some methods that implement it and that have been analyzed in the TFermion software are: Sparsity low-rank [127], Linear T [128] or Taylor naive [129].

8.3 TFermion library

As seen above, quantum algorithms applied to chemistry offer great advantages over classical algorithms. In fact, the only reason why they are not currently used is because there are no quantum devices large enough in number of gates to be executed. Therefore, it is interesting to know what is the real cost in gates of the execution of certain algorithms and to understand if this cost can be reduced by introducing certain optimizations.

In order to obtain detailed data from the analysis of the main state-of-the-art quantum algorithms of GSP and GSEE, a classical software tool has been created following the open source scheme called TFermion. This tool contains the implementation of the T-gate cost of the analyzed algorithms. In this way, the user enters a molecule to be analyzed and the method among those available, and the software returns the cost in T gates.

This gate cost is calculated according to the formulas detailed in the subsection 8.1 and an optimized margin of error according to parameters that can be set by the user in the configuration file. In addition, there are other parameters that can be adjusted, such as the basis functions, the conversion between plane and Gaussian waves, the chemical accuracy, etc.

The effort in quantum computing is divided between an attempt to create algorithms with practical quantum advantage and the ability to run those algorithms on quantum hardware. That leads to lines of research that are more focused on running the algorithms even though there is noise on NISQ-type devices and trying to improve those algorithms. On the other hand, there are lines more focused on developing error correction and fault-tolerant computers. This dichotomy provokes the discussion of how far it is to run certain quantum algorithms on fault-tolerant computers or, seen in another way, what resources a quantum device must have to be suitable for running quantum algorithms.

In that vein, TFermion focuses on making a reliable estimate of how many quantum bridges are needed to run the quantum algorithms that are best positioned to achieve practical quantum advantage, the quantum chemistry algorithms. Software tools such as TFermion are needed to size resource needs, compare lines of research and make technology estimates. Right now, in quantum computing, the quality of the solution, in this case the accuracy with which it simulates a quantum chemistry system, is almost as important as the resources used for it. A high-precision solution may be so expensive to run that it makes no sense to continue its development, or even if it is inexpensive with simple molecules, its scalability is abysmal. All of these metrics can be applied using TFermion.

TFermion can be useful when developing new algorithms and applying them to specific problems. For example, a scientist who wants to implement a quantum algorithm to obtain the ground state energy of a molecule will be able to decide which encoding or which mapping to use based on the results of other algorithms applied to that molecule. Once implemented, he will be able to see how his method compares with the state-of-the-art and know where he has advantages and where he needs to improve. Before TFermion, this possibility did not exist.

In order for TFermion to have continuity and remain valid as research progresses, it has been developed and published following the open-source philosophy. Thus, it is publicly available on GitHub. It is a modular software that is easy to update. Thus, when a scientist develops a new algorithm and wants it to be in TFermion, he can create an issue and ask for it to be included

or, directly, include it in the code. Then, when the user goes to run TFermion, he will see that the list of available methods has been increased.

One of the TFermion use cases to be analyzed in the next chapter 9 is the application of the library to the development of electric batteries. In this case, TFermion allows for a detailed analysis of the cost of the algorithms for a specific problem, of great complexity and with numerous industrial applications. As before, a specific algorithm for the development of a system with quantum properties is analyzed.

In order to follow a fully modular philosophy and to facilitate external collaborations on the TFermion code, each implemented method has been split into different python files. In this way, if a scientist wants to include his own algorithm, he only has to create a concrete class that, respecting a basic structure, calculates the count of T gates of his algorithm. To do this, he must analyze the cost of each of the steps required for its implementation. The extraction of the system configuration to be analyzed, the error optimization, and other technical aspects are automatically performed by TFermion. Once the code is implemented, an administrator is asked to verify it, and, if approved, the new quantum algorithm is incorporated into TFermion.

The gate count for TFermion must be expressed in T gates. T-gates are single-qubit gates that, in combination with other gates, can do what is called universal quantum computation. Therefore, a given result in T gates is as general as possible and serves as an upper bound and can be optimized by using other gates. To understand this, it is necessary to explain some concepts beforehand.

DiVincenzo's criteria [130] assert that for a quantum computer to be scalable, it must demonstrate the ability to execute a universal set of quantum gates. This comprehensive set should encompass all the gates essential for conducting any quantum computation, ensuring that any computation can be expressed as a finite sequence of these universal gates. Fundamentally, a quantum computer, as per these criteria, should not only manipulate individual qubits through single-qubit gates, but also introduce entanglement in the system, a functionality achieved through multi-qubit gates.

Following this definition, two sets of gates can be distinguished, Clifford and non-Clifford. The Clifford gate set consists of Hadamard, phase S, and CNOT gates. This gate set is considered minimal because excluding any one gate renders the inability to implement certain Clifford operations. Specifically, omitting the Hadamard gate eliminates the capability to represent powers of $\frac{1}{\sqrt{2}}$ in the unitary matrix, excluding the phase gate disallows the presence of i in the unitary matrix, and removing the CNOT gate restricts the controlled operations. Given that all Pauli matrices can be constructed from the phase and Hadamard gates, each Pauli gate is inherently an element of the Clifford group.

In addition, a quantum circuit consisting only of gates of the Clifford ensemble can be simulated efficiently. Therefore, to achieve quantum advantage it is necessary to use gates of the non-Clifford set, such as the T-gate.

Therefore, measuring the complexity of running an algorithm on T-gates better represents the bottleneck that exists when running it on quantum hardware. Clifford type gates are more dependent on the topology of the chip and therefore a more variable result. Another important point is that the real limitation of today's quantum computers is not in the number of qubits they have, but in the number of gates, circuit depth, that they can run in quantum coherence and without noise effects.

The gate count performed by TFermion is exclusively adjusted to the operations performed by the quantum algorithm to calculate the ground state energy, leaving aside auxiliary operations such as ground state preparation or the execution by some algorithms of quantum Fourier transform techniques.

Each of the algorithms analyzed in TFermion has a set of parameters that can be varied to obtain the final gate cost. The algorithms allow a chemical accuracy value to be set at which the energy is calculated. In this way, certain parameters of the algorithms can be optimized so that, although the result is always within the desired accuracy, the cost of running the algorithm in gates is minimized. For this purpose, TFermion uses different optimizers that allow to calculate the minimum gate cost T of the algorithm when it is executed with a given accuracy.

In order to test the results of TFermion with a molecule that was used as a standard by other methods and to compare their results, FeMoCo was chosen. It is a molecule with well-known and widely studied properties. Since other authors have also analyzed the cost of applying some of the TFermion algorithms on FeMoCo, this molecule can also be used to validate the TFermion results. FeMoCo was the molecule used to compare the cost of the quantum algorithms used in TFermion.

8.4 Results

- ✓ The use of QROM techniques in the plane wave naïve Taylorization method makes it particularly efficient, and there are indications that employing plane waves could be more effective than Gaussian for the same Taylorization techniques in isolated molecules.
- ✓ TFermion has been created as a software tool to help make decisions about quantum algorithms for quantum chemistry simulations.
- ✓ The main state-of-the-art quantum algorithms for GSP and GSEE have been analyzed, making an exhaustive comparison between them.

- ✓ A quantitative comparison, in terms of implementation cost, between different encoding, mapping and representation methods has been established.
- ✓ A line of hybrid algorithms has been developed following the model of using classical computation as benchmark and assistance of quantum algorithms.
- ✓ It has been shown that the most promising Hamiltonian simulation methods are, on the one hand, to use qubitization, gaussian basis and rank factorization and, on the other hand, Qubitization or Interaction Picture with a plane wave basis and an encoding in first quantization.
- ✓ Algorithm implementation costs have been provided to understand the gate count requirements of quantum algorithms, which is useful in guiding the development of fault-tolerant quantum devices.
- ✓ As shown in figure 3 of 8.4, using TFermion different molecules have been analyzed to each of the studied methods. Initially, simple molecules such as H_2 and HF were analyzed just to test the algorithm. Then, more complex molecules such as NH_4, CO_2 or $NaCl$ were analyzed to show that TFermion is useful.

TFermion: A non-Clifford gate cost assessment library of quantum phase estimation algorithms for quantum chemistry

Pablo A. M. Casares[1], Roberto Campos[1,2], and Miguel A. Martin-Delgado[1,3]

[1]Departamento de Física Teórica, Universidad Complutense de Madrid.
[2]Quasar Science Resources, SL.
[3]CCS-Center for Computational Simulation, Universidad Politécnica de Madrid.

Quantum Phase Estimation is one of the most useful quantum computing algorithms for quantum chemistry and as such, significant effort has been devoted to designing efficient implementations. In this article, we introduce TFermion, a library designed to estimate the T-gate cost of such algorithms, for an arbitrary molecule. As examples of usage, we estimate the T-gate cost of a few simple molecules and compare the same Taylorization algorithms using Gaussian and plane-wave basis.

Contents

Pablo A. M. Casares: pabloamo@ucm.es
Roberto Campos: robecamp@ucm.es
Miguel A. Martin-Delgado: mardel@ucm.es

1 Introduction

Among the different applications found for quantum computing, the original aim of using quantum computers to simulate quantum systems and dynamics [24] still stands out as the most promising one. The reason is twofold: first, a quantum computer can encode the state of the system without needing approximations; and second, since the evolution of (closed) quantum systems is unitary, simulating it is rather natural.

Specifically, quantum computing might be particularly useful to prepare ground states of electronic Hamiltonians and find out their energies. Consequently, they can be employed in a multitude of chemical and material science problems where the ground state energy plays a key role. This includes for instance computing chemical reaction rates [58, 69], and analyzing battery properties [21, 35] or biological enzymes [28].

There exist classical computing techniques able to tackle these problems, most notably Density Functional Theory (DFT) [37]. However, they often rely on approximations, for instance, the Kohn-Sham exchange-correlation parametrized functional, which may struggle to achieve the high accuracy required in some of the problems above. For example, chemical reaction rates depend exponentially on differences in energy. In contrast, the well-known technique Quantum Phase Estimation (QPE) in principle allows achieving the high precision required by these applications. To understand how it works, remember that the Schrodinger equation dictates how a quantum system evolves according to its Hamiltonian,

$$\hat{H}\left|\psi\right\rangle = i\hbar\frac{d}{dt}\left|\psi\right\rangle. \tag{1}$$

If we assume for simplicity that such Hamiltonian is time independent, we can write

$$\begin{aligned}\psi(x,0) &= \sum_n a_n \psi_{E_n}(x) \Rightarrow \\ \psi(x,t) &= \sum_n a_n e^{-iE_n t/\hbar}\psi_{E_n}(x),\end{aligned} \tag{2}$$

for $\psi_{E_n}(x)$ an eigenstate, and E_n the corresponding eigenvalue. We are interested in the ground state energy E_0. Note how the eigenvalues became a phase. To recover it, we can use an inverse Quantum Fourier Transform that will encode such phase in the computational basis, from where it can be readily read out.

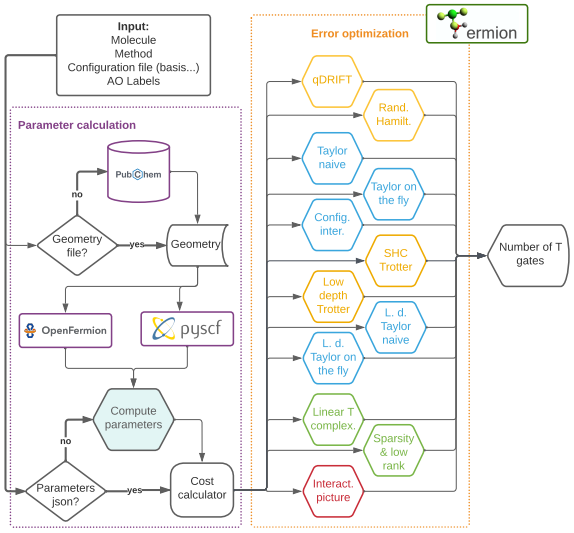

Figure 1: Flowchart of the architecture of our library, divided into two parts: the first one centered on the computation of the parameters needed for the cost estimate; and a second one on using such parameters to compute the number of T-gates. Methods are colored according to the Hamiltonian simulation technique in figure 2.

To implement such an algorithm we need to specify how to implement the quantum Fourier transform, and also the Hamiltonian simulation e^{-iHt}. The former is rather straightforward and can be found in Ref. [53] for example, but the latter is more involved. Furthermore, to obtain a binary description with b bits of the eigenvalue and probability of failure p_f, quantum phase estimation will need to implement $(e^{-iHt})^{2^j}$ for j in the range $1, \ldots, b + \left\lceil \log_2\left(\frac{1}{2} + \frac{1}{2p_f}\right)\right\rceil$ [19]. In other words, it will require the implementation of several time segments that scales with the inverse precision, $O(1/\epsilon_{QPE})$. For this reason, it is important to be able to implement Hamiltonian simulation efficiently.

Such a Hamiltonian might be accessed by the quantum computer in different ways. For electronic Hamiltonians, the most convenient one often is in the form of Linear Combination of Unitaries (LCU). In such framework, we decompose $H = \sum_j a_j H_j$, for a_j some real positive coefficients, and H_j the unitaries, often Pauli string-like operators.

Given such access, there are also various methods to simulate the Hamiltonian evolution. The first way discovered was the Trotter method [2, 42], and soon others such as Taylor series (or Taylorization) [9], Qubitization [43, 45] and Interaction picture simulation (or Dyson series) [34, 44] followed. These Hamiltonian simulation techniques, reviewed in section 3, are the backbone of the quantum phase estimation algorithm. Their objective is to lower as much as possible the computational cost of QPE, so large quantum sys-

tems can be simulated to high precision in reasonable amounts of time, once fault tolerant quantum computers become available. The library we present in this article, TFermion, is an attempt to standardize and automatise the computation of the cost of several quantum phase algorithms in the literature.

However, to use quantum phase estimation, we need to prepare states with a large overlap with the ground state. This will translate into a high probability of measuring the actual ground state energy, and upon success will also project the system into the ground state. Unfortunately, it is known that preparing a representation of the ground state of a 2-body quantum Hamiltonian is Quantum Merlin Arthur (QMA) complete [33], that is, a quantum computer can efficiently verify the solution, but not necessarily efficiently compute it. In other words, finding the ground state of a 2-body Hamiltonian is not in the Bounded Quantum Polynomial-time (BQP) complexity class, the class of problems a quantum computer can solve in polynomial time. Nevertheless, this does not imply either that we cannot propose algorithms to solve it as efficiently as possible [26, 41]. While it is known that the general 2-body ground state preparation is QMA-complete, there is hope that the specificity of electronic Hamiltonians will make it easier to solve at least heuristically. In fact, over the years significant effort has been devoted to the formulation of shallow-depth NISQ ansätze [76] to prepare ground states such as the Imaginary Time Evolution ansatz [47] and the Variational Quantum Eigensolver (VQE) with Unitary Coupled-Cluster [54], adaptative [29], and hardware-efficient [32] ansätze.

Similarly, some effort has been devoted to resource estimates of particular applications [35, 58, 69], but to the best of our knowledge, no software library has been developed to allow a principled comparison between methods. This is a gap that TFermion aims to fill with the following contributions:

First, while newer algorithms often provide a specific non-Clifford gate and qubit count, older ones only give asymptotic complexity estimates (see figure 2). Our article aims to estimate the T-gate cost of older and some of the newer algorithms, with a molecule of choice from the software users. We believe this will be helpful to more quickly carry out research for both academics and industry. Not only that, but our software automatically performs optimization based on the different error sources to minimize the cost, and low-rank approximations [11].

Second, as an example of use of our software, we address the question of whether Gaussian basis functions or plane waves are more convenient to simulate molecules, comparing the same Taylor series algorithms with a different basis. This comparison is not definitive because the error arising from a finite-size basis is difficult to estimate. However, we can give an idea of which algorithms might be more beneficial according to some rough estimates of how many plane waves are required to simulate a system to the same precision than if one were to use Gaussian basis [6]. We furthermore provide researchers with the possibility to carry out a similar comparison but deciding the multiplicative factor for plane waves to represent a similar precision or if the comparison is not the objective, the number of plane waves too.

In TFermion, so far we have focused on T-gates as we believe that non-Clifford gates represent a more significant bottleneck than the number of qubits. Nevertheless, in the future, we expect to add this functionality and additional algorithms to the library. The article itself is structured as follows: first, we give an overview of the library and how it works. Then, in section 3 we briefly explain some of the techniques for Quantum Phase Estimation and Hamiltonian simulation, including figure 2 and table 3 to help the reader understand the development and relation between different algorithms. In section 4, we give examples of how our software might be used, including the second contribution listed above. We then summarize the conclusions and present future work. Finally, in each appendix, we quickly describe one of the techniques studied in this paper, that can be used in combination with the original references to understand the cost estimation functions of TFermion.

2 The TFermion library

The first and main contribution of this article is a software library called TFermion that automatizes the estimation of T-gate cost of running a variety of Quantum Phase Estimation algorithms proposed in the literature during the last years, over arbitrary molecular geometries.

We envision several use cases of our library:

1. It could serve as a quick assessment for the feasibility of concrete QPE experiments once error-corrected quantum computers become available, such as those centered in particular scientific or industrial use cases [35, 58].

2. It can also help make comparisons between systems and methods. In particular, it allows comparing the impact of the chosen Hamiltonian simulation technique, or the chosen basis.

The result provided by our library though must be interpreted as an approximation to the true value, as the final implementation will be heavily optimized, both at a hardware and software level. Our library, in contrast, aims to be more modular and system-agnostic, but we nevertheless provide built-in error optimization. It is well known that different error sources impact the final precision and gate cost in different ways. As such we have aimed at standardizing the way error sources are treated and optimized (see table 1).

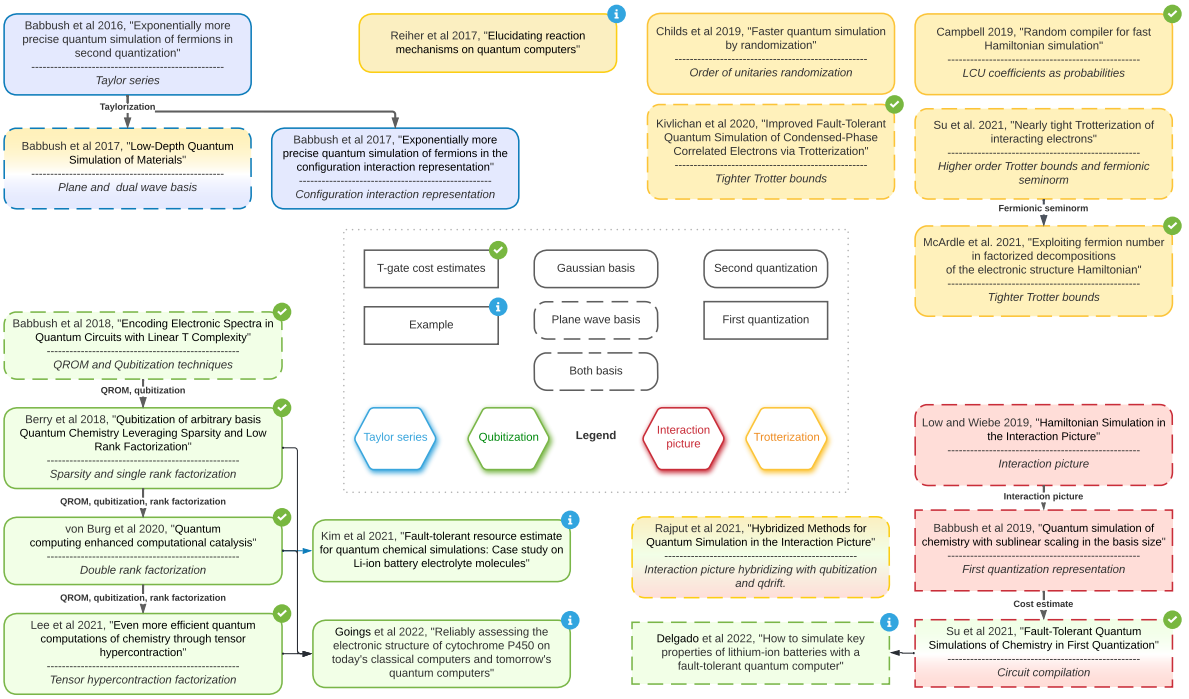

Figure 2: Diagram showing some of the main techniques involved in the development of post-Trotter Quantum Phase Estimation Techniques. Not shown in the picture but of great importance are the articles crystallizing the concept of 'Taylorization' [9] and 'qubitization', [44].

Error	Mathematical definition	Where does it appear?
ϵ_{QPE}	$\epsilon_{QPE} = \lambda 2^{-n}$, n precision bits in the QPE algorithm, and λ the 1-norm of the Hamiltonian.	Due to the Phase Estimation.
ϵ_{HS}	Trotter: $\left\|\left\|e^{-iHt/r} - \mathcal{S}_p(H;t/r)\right\|\right\|_2 \leq W_p \left(\frac{t}{r}\right)^{p+1} \leq \frac{\epsilon_{HS}}{r}$ [48]. Taylor: $\left\|\left\|\Pi_0 A \left\|0\right\rangle \left\|\psi\right\rangle - \left\|0\right\rangle U_r \left\|\psi\right\rangle\right\|\right\|_2 \leq \frac{\epsilon_{HS}}{r}$ [9]. Dyson: $\left\|\left\|W - \mathcal{T}[e^{-i\int_0^{t/r} H(s)ds}]\right\|\right\|_2 \leq \frac{\epsilon_{HS}}{r}$ [44].	In Hamiltonian Simulation via Trotter, Taylor or Dyson series decomposition of $e^{-iH\tau}$.
ϵ_H	$\left\|\int_\Omega f(\mathbf{x})d\mathbf{x} - \sum_{\mathbf{x}\in\Omega} f(\mathbf{x})(\Delta\mathbf{x})^d\right\| < \epsilon_H$, with $d = dim(\Omega)$	Error from the approximation of integrals by Riemannian sums.
ϵ_S & ϵ_{SS}	$\|\|U - R_z(\theta)\|\|_2 \leq \epsilon_{SS}$ [60] (Using operator norm)	In the synthesis of single rotations ϵ_{SS} and their sum, ϵ_S.
ϵ_{tay}	Defined as in Taylor's theorem.	Due to Taylor error series (and others) in arithmetic operations.

Table 1: Notation for the main sources of error that we take into account in the article and software library. Additional minor sources may appear sporadically in single articles. The norm 2 used in all cases above is the operator norm. The other algorithms used to compute arithmetic operations are the Babylon algorithm for the square root, and CORDIC algorithm for the sine or cosine. $\mathcal{S}_p(H;t/r)$ stands for the order p Trotter step, and $W_p = O\left(\max_{\boldsymbol{i}}[[\ldots[H_{\gamma_{i_1}}, H_{\gamma_{i_2}}], H_{\gamma_{i_3}}], \ldots], H_{\gamma_{i_{p+1}}}]\right)$ the commutator terms that bound the final error [63].

While not the main objective of our article, we also believe our work may help provide a more standardized treatment across methods, and as a consequence help better understand the choices in the Hamiltonian simulation, basis, or fermion-qubit mapping used.

One feature of our library is that it currently contains older than 1-year-old methods, and as such some excellent work [39, 62, 69] has not yet been included. There are two reasons: the first and most obvious one is that including new methods represents a significant amount of effort, and we believe these updates can be done later on. The second is that while for the latest methods T-gate estimates are more common, for older ones often only the complexity estimates are available. While this makes sense as the latest methods might be more useful for industrial processes, we believe that understanding well different techniques and not only the bleeding edge ones can be of significant scientific interest.

Additionally, our software was developed following a modular architecture with an easy procedure to include new methods. The process to add a newer method or updating an existing one requires two main steps: first making sure that the molecular parameters required are already calculated by some of the provided methods, or adding new ones in `molecule.py`; then create a new T-gate cost estimation function and call it from the class `Cost calculator`. The philosophy underlying this architecture is to keep TFermion updated timely and give the authors of the new methods the possibility to add their own T gate cost estimation to show practical examples of their work and make it more accessible.

The use of the library is rather straightforward: the user only needs to provide a `molecule name`, a `method` and optionally some atomic orbital labels (`ao labels`) to be used within the active space selection method AVAS [59] to restrict the calculation and make it more efficient. This should be supplemented within a configuration file with the Gaussian basis to be used. If the method requires plane waves to be used, the system will by default approximate the number of basis functions as the thumb rule of 100 times more plane waves than Gaussian waves [6]. Alternatively, the user might provide this and other molecular parameters (eg λ, N...) in a JSON file under the name `[molecule name]_[basis].json`. A flowchart of the working of the library can be seen in figure 1.

As it is shown in figure 1, TFermion is executed through a `main` module which receives the molecule name, the QPE method, and optionally also the ao-labels to select an active space using AVAS. It starts with the `molecule` module creating a `molecule` instance, which is passed together with the method to `cost calculator`. The latter one calls either Gaussian or Plane Waves `molecule` methods to calculate all necessary parameters. Finally, `cost calculator` minimizes the cost depending on the error sources on the selected method, and sends the result back to `main`.

Operation	Cost
Addition & subtr. [27]	$4n$
Multiplication [52]	$21n^2$
Division [67]	$14n^2 + 7n$
Comparison [20]	$8n$
Multi-controlled Not [8]	$16(m-2)$ m controls
Rotation synthesis [60]	$10 + 12\lceil \log_2 \epsilon_{SS}^{-1} \rceil$, $SU(2)$ $10 + 4\lceil \log_2 \epsilon_{SS}^{-1} \rceil$, R_z
State synthesis [61]	$2^{n+1} - 2$ arbitrary rotations

Table 2: Cost of basic arithmetic operators in T gates unless otherwise stated, omitting additive $O(1)$ factors. If the rotation synthesis is controlled, the cost will be multiplied by 2 for R_x, R_y and R_z gates, as given by Lemma 5.4 in [8]. Notice that $HR_zH = R_x$, while R_y and R_z are particular cases of the unitary W in that Lemma. Finally, for general controlled rotations the cost will be thrice the synthesis cost instead of twice.

TFermion manages four types of data:

- **Molecule:** A class created to save all the molecular data, including geometric information obtained [12] used to compute the electronic integrals using Pyscf [64].
- **MolecularData:** An instance from the OpenFermion class [49], necessary to get all parameters from the Hamiltonian and save them into instance `molecule` as attributes.
- **Error values:** Different QPE methods have different error sources, whose sum must not exceed a given threshold. By default we will use the `chemical accuracy` value of 0.0016 Hartrees [17]. TFermion optimizes error values to minimize the T-gate cost output of that method without exceeding it.
- **T gate cost:** Number of T gates needed to execute the selected method, as well as the time required to synthesize the corresponding number of magic states. Calculating this value is the main goal of our library.

Certain calculations in the library are computationally and memory intensive. The reason for this is that as the number of basis functions grows, so does the size of the one and two-body Hamiltonian terms, but does so at least quadratically. This is reflected especially in the plane wave case for molecules, where the larger number of plane waves is due to the need for significantly more basis functions. Nevertheless, an effort has been put into making the calculations relatively efficient, making use of some new techniques [38].

Algorithm	Simulation	Quantization	Basis	Encoding
Random Hamiltonian [15, 18]	Trotter	2nd quantization	Gaussian	Jordan-Wigner
qDRIFT [14, 15]	Trotter-related	2nd quantization	Gaussian	Jordan-Wigner
Taylorization 'database' [3]	Taylor series	2nd quantization	Gaussian	Jordan-Wigner
Taylorization 'on-the-fly' [3]	Taylor series	2nd quantization	Gaussian	Jordan-Wigner
Configuration Interaction [4]	Taylor series	1st quantization	Gaussian	Slater determinant
Low-depth 'Trotter' [6]	Trotter	2nd quantization	Plane waves	Jordan-Wigner
Low-depth 'Taylor database' [6]	Taylor series	2nd quantization	Plane waves	Jordan-Wigner
Low-depth 'Taylor on-the-fly' [6]	Taylor series	2nd quantization	Plane waves	Jordan-Wigner
Interaction picture [45]	Dyson series	2nd quantization	Plane waves	Jordan-Wigner
Sublinear scaling inter. pict. [7, 62]	Dyson series	1st quantization	Plane waves	Slater determinant
Sublinear scaling qubitization [7, 62]	Qubitization	1st quantization	Plane waves	Slater determinant
Linear T complexity [5]	Qubitization	2nd quantization	Plane waves	Jordan-Wigner
Sparsity and low rank [11]	Qubitization	2nd quantization	Gaussian	Jordan-Wigner
Double factorization [69]	Qubitization	2nd quantization	Gaussian	Jordan-Wigner
Tensor hypercontraction [39]	Qubitization	2nd quantization	Gaussian	Jordan-Wigner
Hybridized method [57]	Trotter & Dyson	2nd quantization	Plane waves	Jordan-Wigner

Table 3: Recent Hamiltonian simulation methods, named after the techniques they use, or the title of the corresponding article, explaining them for efficient Hamiltonian simulation and Quantum Phase Estimation. Notice that qDRIFT, Random Hamiltonian and Hybridize method do not specify the basis or the Fermionic encoding, but the ones we indicate seem to be the most obvious: in the case of qDRIFT and Random Hamiltonian because they are the simplest choice, while in the Hybridized method, it inherits the plane wave structure from the Interaction Picture. Recent work on Trotter Hamiltonian simulation [15, 36, 48, 62] has focused on bounding commutator error terms on a different basis, rather than new methods.

Finally, let us briefly mention what our software does not cover yet. It only provides cost estimates for T-gate count, as it is well known that the magic state distillation required to perform the T-gate often carries the largest cost in 2 the dimensional surface code, which nevertheless exhibits a large threshold. Alternatively, there are codes in 3D, like topological color codes [13] that avoid magic state distillation, and may provide new ways to improve this counting, but they require more qubits for similar distance codes. Furthermore, the Clifford gate count may depend on the specific chip connectivity, and for that reason, we have preferred to ignore it here. Finally, while we believe that the qubit count is important, the number of gates required may provide a more significant constraint in the long term due to the time required to perform the algorithms, as these approaches usually require on the order of 10^2 to 10^4 qubits for realistic targets [35, 58, 62].

The cost of ground state preparation, while significant, is left for future work too. Rough estimates may be possible to obtain for moderately sized systems, using low precision QPE to project the system into the ground state [10].

3 Quantum phase estimation techniques

In this section, we give a quick overview of the main techniques used in the literature to perform quantum phase estimation. Quantum phase estimation requires two main ingredients: the use of a controlled Hamiltonian simulation method and sometimes an inverse Quantum Fourier Transform (QFT). While the original Quantum Phase Estimation protocol did use QFT [25, 53], more modern versions such as Bayesian Quantum Phase Estimation avoid it [75]. This latter approach has also the property of being parallelizable, implementable with minimal classical postprocessing, and requires fewer qubits. However, its cost scales as $\frac{4.7\lambda}{\epsilon_{QPE}}$ instead of the theoretical optimum of $\frac{\pi\lambda}{\epsilon_{QPE}}$ [75]. Since the extra cost of the quantum Fourier transform and the qubits it requires are often negligible, we will instead assume we are using the classical version with a slightly lower cost. We will now explain the other main part, the different Hamiltonian simulation techniques.

3.1 Trotter

Let us assume we want to simulate H for a Linear Combination of Unitaries decomposition $H = \sum_\gamma w_\gamma H_\gamma$. The difficulty is that since the different unitaries H_γ do not need to commute, we cannot write $e^{-iHt} = \prod_\gamma e^{-iw_\gamma H_\gamma t}$. Instead, using the product of

Hamiltonian simulation as we have just done introduces an error $O(\sum |[H_{\gamma_1}, H_{\gamma_2}]|t^2)$ that depends on the commutator.

To handle this error, within the scheme of Trotter, there are two strategies. The first one is to divide the evolution in short time segments so we can quadratically suppress the error. In other words, we implement

$$e^{-iHt} = \left(\prod_{\gamma} e^{-iw_\gamma H_\gamma t/r}\right)^r + O\left(\sum |[H_{\gamma_1}, H_{\gamma_2}]|t^2/r\right). \tag{3}$$

Alternatively, one may attempt to find higher order Trotter formulas that further suppress the error. For example, if (3) is the first order formula, then

$$e^{-iHt} = \left(\left(\prod_{\gamma=1}^{\Gamma} e^{-iw_\gamma H_\gamma t/2r}\right)\left(\prod_{\gamma=\Gamma}^{1} e^{-iw_\gamma H_\gamma t/2r}\right)\right)^r + O\left(\sum |[[H_{\gamma_1}, H_{\gamma_2}], H_{\gamma_3}]|t^3/r^2\right) \tag{4}$$

is the second order one. Higher-order formulas are known, but they also become more convoluted to implement. Another possibility is to use classical randomization of the order in which each of $w_\gamma H_\gamma$ appears in the Hamiltonian, in each evolution segment [18], or to apply Hamiltonian simulation of a random H_γ for fixed amounts of time, with probabilities given in by w_γ/λ for $\lambda = \sum w_\gamma$ [15]. The latter method is called 'qDRIFT' and is explored in appendix A together with a second-order randomized Trotter simulation. Other randomized methods have been explored too [70].

There has also been effort devoted to tightly bounding the commutators to reduce the number of segments [16, 36, 48, 63]. Of these, one with a favorable scaling number of basis functions, $O(N^3)$, is the so-called 'SHC bound' for dual wave basis Hamiltonian [48, 63]. It is implemented as the method `shc_trotter` in our library and can be found in appendix F. Finally, Trotter simulation has historically been one of the first methods to be used to estimate resource estimates, including the famous FeMoco study [58], and later ones [22].

3.2 Taylor series

Methods invented after Trotterization are usually called post-Trotter, and their objective is to lower the Hamiltonian simulation error dependence, ϵ_{HS}, from polynomial to polylogarithmic. Taylor series simulation or Taylorization aims to expand the evolution operator of a small time segment as a Taylor series

$$U_r = e^{-iHt/r} \approx \sum_{k=0}^{K} \frac{1}{k!}(-iHt/r)^k = \sum_{k=0}^{K}\sum_{l_1,\ldots,l_k=1}^{L} \frac{(-it/r)^k}{k!} a_{l_1}\ldots a_{l_k} H_{l_1}\ldots H_{l_k}. \tag{5}$$

This expression is a Linear Combination of Unitaries (LCU), $U_{LCU}^{\text{Tay}} = \sum_{l=0}^{L} b_l U_l$. To implement it, one introduces operators

$$\text{Prepare} : |0\rangle \mapsto \sum_l \sqrt{b_l}\,|l\rangle\,, \tag{6}$$

$$\text{Select} : |l\rangle\,|\psi\rangle \mapsto |l\rangle\,U_l\,|\psi\rangle\,, \tag{7}$$

and defines $U_{LCU}^{\text{Tay}} = (\text{Prepare}^\dagger \otimes \mathbf{1})\text{Select}(\text{Prepare} \otimes \mathbf{1})$. Since U_{LCU}^{Tay} has some failure probability in recovering $|0\rangle$ in the first register, it is customary to use (oblivious) amplitude amplification [9], that reduces the error to ϵ_{HS}/r in each segment.

3.3 Block encoding and qubitization

Similarly, the Hamiltonian often takes the form of a linear combination of unitaries $H = \sum a_l H_l$, from which we can create as the block-encoding operator

$$U_{LCU} = \begin{pmatrix} H/\lambda & \cdot \\ \cdot & \cdot \end{pmatrix}, \tag{8}$$

with new Prepare and Select operators

$$\text{Prepare} : |0\rangle \mapsto \sum_l \sqrt{a_l}\,|l\rangle\,, \tag{9}$$

$$\text{Select} : |l\rangle\,|\psi\rangle \mapsto |l\rangle\,H_l\,|\psi\rangle\,. \tag{10}$$

Using them, we obtain,

$$U_{LCU}\,|0\rangle\,|\psi\rangle = |0\rangle\,\frac{H}{\lambda}\,|\psi\rangle + \sqrt{1 - \frac{\|H\,|\psi\rangle\,\|}{\lambda}}\,|(0,\psi_\lambda)^\perp\rangle\,. \tag{11}$$

However, as we saw this LCU implementation has some probability of failure, which requires amplitude amplification to suppress. An alternative is to construct a quantum walk operator Q with the same spectrum. This is done via a procedure called qubitization [45]. In the case where the corresponding $U^2 = 1$, as is the case for $U_{LCU} = \text{Prepare}^\dagger \cdot \text{Select} \cdot \text{Prepare}$, it can simply be implemented as [45, Corollary 9]

$$Q = \underbrace{\text{Prepare}(2\,|0\rangle\,\langle 0| \otimes \mathbf{1} - \mathbf{1})\text{Prepare}^\dagger}_{R} \cdot \text{Select}. \tag{12}$$

Q implements a Grover rotation in each eigenspace

$$\begin{aligned} Q\,|0\rangle\,|\psi_k\rangle &= \cos(\theta_k)\,|0\rangle\,|\psi_k\rangle - \sin(\theta_k)\,|(0,\psi_k)^\perp\rangle\,, \\ Q\,|(0,\psi_k)^\perp\rangle &= \cos(\theta_k)\,|(0,\psi_k)^\perp\rangle + \sin(\theta_k)\,|0\rangle\,|\psi_k\rangle\,, \end{aligned} \tag{13}$$

for $\cos\theta_k = \frac{E_k}{\lambda}$. In other words, Q is a quantum walk operator

$$Q = \bigoplus_k \begin{pmatrix} \frac{E_k}{\lambda} & -\sqrt{1-\frac{E_k^2}{\lambda^2}} \\ \sqrt{1-\frac{E_k^2}{\lambda^2}} & \frac{E_k}{\lambda} \end{pmatrix}_k . \tag{14}$$

Diagonalizing the subspace spanned by $\{|0\rangle|\psi_k\rangle, |(0,\psi_k)^\perp\rangle\}$, we might write $Q_{LCU} = \bigoplus_k \left(e^{i\theta_k}|\theta_k\rangle\langle\theta_k| + e^{-i\theta_k}|-\theta_k\rangle\langle-\theta_k|\right)$. We can use this operator to create a Chebyshev series that approximates e^{-iHt} [44], with a technique called quantum signal processing [43]. However, it is more straightforward to apply phase estimation directly over $\pm\theta_k$ [10]. Then, computing $\cos(\theta_0)$ we recover the ground state energy.

Additionally, qubitization has the advantage that $RQR = Q^\dagger$, so using this trick we can duplicate the implemented phase with almost no extra cost, so the prefactor in the cost falls from $\frac{\pi\lambda}{\epsilon_{QPE}}$ to $\frac{\pi\lambda}{2\epsilon_{QPE}}$ [5]. Qubitization is often used in combination with QROM and factorization techniques [5, 11, 39, 69], but has also been used in first quantization [7, 62].

3.4 Interaction picture and Dyson series

While the qubitization method is optimal concerning the Hamiltonian simulation error, an alternative approach is to find ways to decrease the 1-norm λ of the Hamiltonian H. Let us assume that $H = A + B$ such that $\|A\| \gg \|B\|$. In the interaction picture, $H_I(t) = e^{iAt}B(t)e^{-iAt}$, so in this framework, the norm of the Hamiltonian decreases to $\|B\|$, and therefore the phase estimation may be cheaper to implement. In this picture, the Hamiltonian simulation is implemented as

$$|\psi(t)\rangle = e^{-iAt}\mathcal{T}\left[e^{-i\int_0^t H(s)ds}\right]|\psi(0)\rangle, \tag{15}$$

where $\mathcal{T}$ denotes time ordering. While the e^{-iAt} might be easy to implement if all unitary operators in LCU decomposition of A commute, the time ordered exponential is more difficult to implement. This might be done with a Dyson series

$$\begin{aligned} U(t) &= \mathcal{T}\left[e^{-i\int_0^t H(s)ds}\right] = \sum_{k=0}^{\infty}(-i)^k D_k \\ D_k &= \frac{1}{k!}\int_0^t \ldots \int_0^t \mathcal{T}[H(t_k)\ldots H(t_1)]d^k t, \end{aligned} \tag{16}$$

that similarly to the Taylor series approach, bears a logarithmic complexity on ϵ_{HS}, and requires to implement the simulation for short time segments and use amplitude amplification at each of them. Operator B is implemented as

$$\frac{B}{\|\lambda_B\|} = \langle 0|\text{Prepare}_B^\dagger \cdot \text{Select}_B \cdot \text{Prepare}_B|0\rangle \tag{17}$$

Using this block encoding of operator B, we can express the block encoding of a time segment of $e^{-i(A+B)\tau}$ as [62]

$$\begin{aligned} e^{-i(A+B)\tau} \approx e^{-iA\tau} \lim_{\substack{K\to\infty \\ M\to\infty}} \sum_{k=0}^{K} \frac{(-i\tau)^k}{M^k k!} \sum_{m_1=0}^{M-1} \cdots \sum_{m_k=0}^{M-1} \\ \left(e^{-i\tau(-1/2-m'_k)A/M} B e^{-i\tau(m'_k-m'_{k-1})A/M} B \ldots \right. \\ \left. B e^{-i\tau(m'_2-m'_1)A/M} B e^{-i\tau(m'_1+1/2)A/M}\right) \\ = \left(\langle 0|\,\text{Prepare}_B^\dagger\right)^{\otimes K} \sum_{k=0}^{K} \frac{(-i\lambda_B\tau)^k}{M^k k!} \sum_{m_1,\ldots,m_k=0}^{M-1} \\ \left(e^{-i\tau(M-1/2-m'_k)A/M}\text{Select}_B e^{-i\tau(m'_k-m'_{k-1})A/M} \right. \\ \text{Select}_B \ldots \text{Select}_B e^{-i\tau(m'_2-m'_1)A/M}\text{Select}_B \\ \left. e^{-i\tau(m'_1+1/2)A/M}\right)\left(\text{Prepare}_B|0\rangle\right)^{\otimes K}, \end{aligned} \tag{18}$$

where $m'_1, \ldots, m'_k$ are the sorted integers from $m_1, \ldots, m_k$. This series might therefore be implemented in a similar fashion as those from Taylor series, and will similarly require amplitude amplification. The Dyson series simulation was first introduced in Refs. [34, 45].

4 Results and an use case example: comparison between different basis functions

In this section, we make use of our library to show usage examples. For that purpose, we will perform two tasks: (1) using the FeMoco Hamiltonian provided in the supplementary material of [39], compute the cost of performing Quantum Phase Estimation with several methods included in the library; and (2) perform T-gate estimation for a few simple molecules with a wide range of methods, making a preliminary comparison of the impact of Gaussian or plane-wave basis in the final T gate count, when using Taylorization as a Hamiltonian simulation method.

FeMoco estimates

Over the last years, FeMoco became a standard benchmark for quantum algorithms [58]. Such a benchmark is realistic and useful because it constitutes the metal active center of an enzyme capable of converting atmospheric nitrogen and hydrogen into ammonia, bypassing the energy-intensive industrial Haber-Bosch process. As the first use case example of our library, we first extend the T-gate cost estimation for several methods. Not only this will help us understand the complexity of previous examples, but will also help check the validity of our results for

FeMoco active space	Reiher et. [58]	Li et. [40]
qDRIFT [15]	7.34e+23	3.62e+23
Rand. Hamilt. [15]	1.32e+28	2.94e+28
Taylor naïve [3]	1.15e+22	1.26e+23
Spars. low-rank [11]	2.36e+13	2.17e+13
w/o failure [11]	4.57e+12	4.12e+12
Results in [11]	4.8e+12	3.9e+12

Table 4: Estimation of number of T-gates required to run different Quantum Phase Estimation algorithms with several algorithms. The second half of the table shows that our library gets similar results as [11], where the 'w/o failure' row indicates we obtain without taking into account failure probability.

the low-rank decomposition method, where previous estimates were available [11].

Using the Taylorization approach [3] has intermediate cost between that of Trotterization (qDRIFT and Random Hamiltonian simulation [15]) and more recent rank-decomposition and qubitization techniques [11]. Furthermore, the last row of table 4 can be compared with the published costs of $1.2{\cdot}10^{12}$ and $9.8{\cdot}10^{11}$ Toffoli gates for both active spaces [11, 40, 58]. Since each Toffoli gate is equivalent to 4 T-gates, our estimation is very close to the numbers originally reported. We believe the small difference is due to a combination of factors. In the first place, the error optimization will usually give more weight to ϵ_{QPE} as it is the most costly error source. Additionally, we take into account some factors such as the Uniform subroutines and an amplitude amplification step in the preparation of uniform superpositions on registers

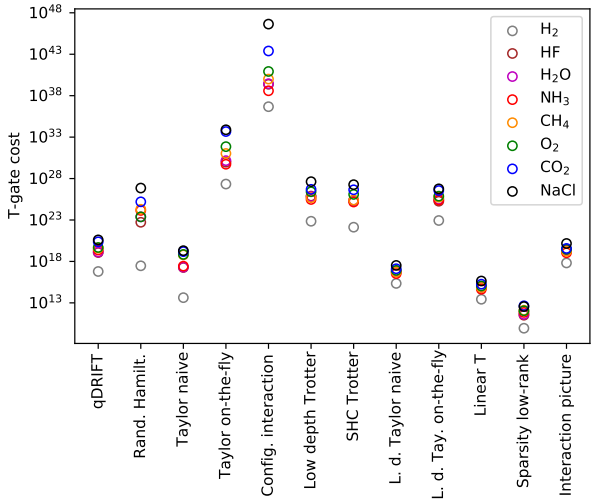

Figure 3: Representation of the results obtained for simple molecules with the results from table 6. We can see that choosing the right method greatly impacts the final cost of the Quantum Phase Estimation algorithm.

N	λ	TFermion	[5] conditions	[5] results
54	5	7.08e+08	2.69e+07	1.80e+07
128	23	4.78e+09	2.26e+08	1.90e+08
250	64	1.96e+10	1.09e+09	1.10e+09
1024	640	5.58e+11	3.88e+10	4.30e+10

Table 5: Replication of the T-gate cost estimates of the `linear_t` method with Jellium, similar to those published in table III from [5]. The third column includes the results with our library, while the fifth those from the original reference [5]. Most of the divergence can be explained because the total error budget has to be allocated between ϵ_{QPE} and ϵ_S, and by considering negligible the rotation synthesis cost. To account for this, the fourth column indicates the results we get if fixed the phase estimation error to $\epsilon_{QPE} = 0.0016$ Hartree, and did not take into account the cost of gate rotation synthesis or failure probability. After this we still do not get the exact results due to other polylogarithmic contributions that the original reference did not considered; but we get quite close.

p and q such that $p \leq q < N/2$ (respectively r and s). We also take a slightly larger number of segments r as described in section 3A of [62], due to the estimation of the phase of $e^{-i\tau \arccos H}$ instead of $e^{-i\tau H}$.

The FeMoco cost of other methods implemented in the library has not been computed, due to the lack of geometry-dependent parameters such as the position of the atoms in FeMoco, or because they were conceived for plane waves instead of gaussian wave functions. In any case, we believe that these results confirm the usefulness of TFermion.

Simple molecules

Next, we run T-gate cost estimates of all the algorithms included in TFermion, with several molecules. As a use-case example, we compare the costs of similar methods on a different basis, something not previously been done in the literature. While these simple molecules can also be analyzed with classical methods, we selected these simple molecules to avoid performing active space selection on them. Of course, selecting such active space in a molecule of scientific interest will represent an important step to making the simulation efficient, but our aim here is to compare the methods rather than obtain novel results for applications of scientific or industrial interest.

The results from our calculations can be seen in table 6. We indicate the median value obtained for each entry after running the procedure 10^3 times. We select the median instead of the average because the results have some inherent stochasticity due to the error sources optimization, but the distribution tends to be skewed to the higher values. We also do not take the lowest value to avoid numerical instability in the ϵ values that may have given rise to unrealistic lower costs.

Method	H_2	HF	H_2O	NH_3	CH_4	O_2	CO_2	NaCl
qDRIFT [15]	6.2e+16	1.2e+19	1.4e+19	2.4e+19	3.9e+19	5.0e+19	2.4e+20	4.0e+20
Rand. Hamilt. [15]	3.0e+17	5.2e+22	2.4e+23	1.4e+24	1.9e+24	2.4e+23	1.6e+25	7.1e+26
Taylor naive [3]	3.0e+13	1.3e+17	1.4e+17	1.9e+17	4.1e+18	4.7e+18	1.1e+19	1.4e+19
Taylor on-the-fly [3]	1.4e+27	5.9e+29	9.4e+29	3.3e+29	6.8e+30	4.6e+31	3.0e+33	4.8e+33
Config. interaction [4]	1.6e+36	2.4e+39	2.8e+39	3.9e+38	1.0e+40	8.3e+40	2.5e+43	4.3e+46
Low depth Trotter [6]	1.2e+23	1.3e+26	1.1e+26	5.0e+25	8.5e+25	4.4e+26	8.4e+26	6.9e+27
SHC Trotter [6, 48]	2.3e+22	3.6e+25	4.2e+25	2.5e+25	4.2e+25	2.0e+26	7.5e+26	3.2e+27
L. d. Taylor naive [6]	3.1e+15	7.8e+16	8.4e+16	4.9e+16	7.6e+16	1.2e+17	1.8e+17	4.7e+17
L. d. Tay. on-the-fly [6]	1.3e+23	2.7e+25	4.7e+25	3.7e+25	8.4e+25	1.1e+26	5.2e+26	8.5e+26
Linear T [5]	3.9e+13	1.0e+15	1.1e+15	6.3e+14	9.7e+14	1.6e+15	2.6e+15	6.3e+15
Sparsity low-rank [11]	**1.2e10**	**4.6e11**	**6.0e11**	**1.0e12**	**1.8e12**	**1.5e12**	**6.3e12**	**5.3e12**
Interaction picture [45]	1.4e+18	5.7e+19	5.0e+19	2.4e+19	3.6e+19	6.6e+19	8.0e+19	3.3e+20

Table 6: T-gate cost estimates for different molecules and methods obtained using our TFermion, see Fig. 3. The Rank decomposition technique is the most efficient between the analysed methods, closely followed by the plane wave methods using QROM and qubitization ('Linear T') or Taylorization ('Low depth Taylor naïve').

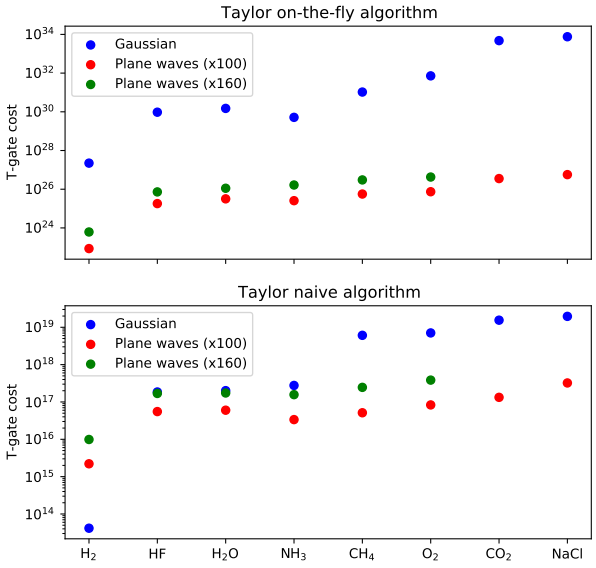

Figure 4: T-gate cost of performing the same algorithms making use of Taylorization as the main Hamiltonian simulation technique, over different molecules. The number of plane waves was chosen to be ≈ 100 or 160 times larger than Gaussian functions as recommended by Appendix E in [6]. The cost of computing the electronic integrals on-the-fly is larger than classically precomputing and loading them. The comparison between Gaussian and plane-wave basis should be taken with care as the error due to finite basis size was not rigorously computed and controlled.

Let us first comment on the results of some methods. The first thing that calls our attention is the large cost of the Configuration Interaction method [4]. We believe this is due to a combination of three factors: the first and most important one is that the condition on the number of segments r imposed by the Lemmas 1-3 in [4] is a very large value, which may be understood as an upper bound rather than a real cost estimate. Secondly, our method to perform the procedure from section 4.1 was not optimized. And thirdly, it also contains a large number of arithmetic operations, similar to those in 'Taylor on-the-fly'. Overall this indicates that the estimates for this method should be treated as an upper bound.

We can also observe that when using a Gaussian basis, Taylor methods are almost always more efficient than Trotter ones and that the cost of using the on-the-fly versions of Taylor is often larger than the naïve one due to the arithmetic operations. The interaction picture algorithm [6] displays a 'similar' complexity as the Taylorization algorithms [3], as both operate on a Gaussian basis and decompose the evolution operator in a Taylor or Dyson series.

The most efficient algorithms among the analyzed ones are those making use of the QROM techniques, [5, 11]. Surprisingly though, the Low depth Taylor naïve [6] achieves the third-best complexity just after the rank-decomposition algorithm [11], and the original article introducing the QROM [5]. We believe the reason for that is that the original article left unspecified the techniques that should be used to implement Prepare and Select, so we have assumed the use of modern QROM techniques [5].

To make this comparison fair, we have, as a rule of thumb, used approximately 100 times as many plane waves as Gaussian wave functions, as it has been suggested for isolated molecules [6]. The Gaussian basis used is the standard 6-31G [31], but this may be changed by the user at will in the configuration file, as well as the multiplicative factor. Using the previously mentioned ratio, we can as an example of usage of our library, compute the cost of the same Taylorization methods with Gaussian and plane waves. The results are shown in figure 4, although these results must be taken with care as we have not controlled the

error introduced by different finite basis sets.

5 Conclusions and future work

Over the last years significant effort has been devoted to creating efficient algorithms for Quantum Phase Estimation and Hamiltonian simulation since the estimation of ground state energy is such a central problem for quantum chemistry and a very natural application of quantum computing. TFermion fills a gap in standardizing and easing the use of such algorithms. It should help academics have a better understanding of algorithms for which no complexity estimates were previously available. The usefulness for the industry is clear too, as it reduces the effort required to quickly iterate over specific use-cases. As examples of usage, we have run calculations with FeMoco and a range of molecules. Among the most interesting results is the fact that using QROM techniques in the plane wave naïve Taylorization method [6] makes it particularly efficient, and we have seen hints that using plane-wave could be more efficient than Gaussian for the same Taylorization techniques in isolated molecules.

However, the effort is far from complete. On one hand, exciting avenues of research remain open, particularly in the use of plane waves [62]. On the other, we aim to improve this library in several dimensions: (1) newer algorithms should be added; (2) our algorithms are designed for molecules instead of materials, where plane-wave methods should become very efficient; (3) TFermion only provides estimates for T-gates so the addition of other metrics such as the number of qubits would be a welcomed addition; and (4) the topic of ground state preparation is barely touched upon but should be considered a prerequisite to estimate the ground state energy.

We believe this is a particularly exciting time to explore how quantum computing can be applied to chemistry and material science. For this reason, we humbly hope that TFermion will become a useful tool to advance the field and find beneficial applications for society.

Code availability

The code for this article can be found at `https://github.com/PabloAMC/TFermion`.

Acknowledgements

We want to thank the very kind explanations of Emiel Koridon of some calculations in one of his articles and beyond. Similarly, we thank answers from Nicolas Rubin and Ryan Babbush on the use of OpenFermion, Joonho Lee on the code from [39], and Antonio Hidalgo, María Jesús Morán, Nelaine Mora and Javier García on quantum chemistry. We acknowledge financial support from the Spanish MINECO grants MINECO/FEDER Projects FIS 2017-91460-EXP, PGC2018-099169-B-I00 FIS-2018, from CAM/FEDER Project No. S2018/TCS-4342 (QUITEMAD-CM), and from Spanish MCIN with funding from European Union NextGenerationEU (PRTR-C17.I1) and Ministry of Economic Affairs Quantum ENIA project. The research of M.A.M.-D. has been partially supported by the U.S. Army Research Office through Grant No. W911NF-14-1-0103. P. A. M. C. thanks the support of a MECD grant FPU17/03620, and R.C. the support of a CAM grant IND2019/TIC17146.

References

[1] Daniel S Abrams and Seth Lloyd. Simulation of many-body fermi systems on a universal quantum computer. *Physical Review Letters*, 79(13):2586, 1997. DOI: https://doi.org/10.1103/PhysRevLett.79.2586.

[2] Alán Aspuru-Guzik, Anthony D Dutoi, Peter J Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707, 2005. DOI: https://doi.org/10.1126/science.1113479.

[3] Ryan Babbush, Dominic W Berry, Ian D Kivlichan, Annie Y Wei, Peter J Love, and Alán Aspuru-Guzik. Exponentially more precise quantum simulation of fermions in second quantization. *New Journal of Physics*, 18(3):033032, 2016. DOI: https://doi.org/10.1088/1367-2630/18/3/033032.

[4] Ryan Babbush, Dominic W Berry, Yuval R Sanders, Ian D Kivlichan, Artur Scherer, Annie Y Wei, Peter J Love, and Alán Aspuru-Guzik. Exponentially more precise quantum simulation of fermions in the configuration interaction representation. *Quantum Science and Technology*, 3(1):015006, 2017. DOI: https://doi.org/10.1088/2058-9565/aa9463.

[5] Ryan Babbush, Craig Gidney, Dominic W Berry, Nathan Wiebe, Jarrod R McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. Encoding electronic spectra in quantum circuits with linear t complexity. *Physical Review X*, 8(4):041015, 2018. DOI: https://doi.org/10.1103/physrevx.8.041015.

[6] Ryan Babbush, Nathan Wiebe, Jarrod R McClean, James McClain, Hartmut Neven, and Garnet Kin-Lic Chan. Low-depth quantum simulation of materials. *Physical Review X*, 8(1):011044, 2018. DOI: https://doi.org/10.1103/physrevx.8.011044.

[7] Ryan Babbush, Dominic W Berry, Jarrod R McClean, and Hartmut Neven. Quantum simulation of chemistry with sublinear scaling in basis size.

npj Quantum Information, 5(1):1–7, 2019. DOI: https://doi.org/10.1038/s41534-019-0199-y.

[8] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457, 1995. DOI: https://doi.org/10.1103/PhysRevA.52.3457.

[9] Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. Simulating hamiltonian dynamics with a truncated taylor series. *Physical Review Letters*, 114(9):090502, 2015. DOI: https://doi.org/10.1103/physrevlett.114.090502.

[10] Dominic W Berry, Mária Kieferová, Artur Scherer, Yuval R Sanders, Guang Hao Low, Nathan Wiebe, Craig Gidney, and Ryan Babbush. Improved techniques for preparing eigenstates of fermionic hamiltonians. *npj Quantum Information*, 4(1):1–7, 2018. DOI: https://doi.org/10.1038/s41534-018-0071-5.

[11] Dominic W Berry, Craig Gidney, Mario Motta, Jarrod R McClean, and Ryan Babbush. Qubitization of arbitrary basis quantum chemistry leveraging sparsity and low rank factorization. *Quantum*, 3:208, 2019. DOI: https://doi.org/10.22331/q-2019-12-02-208.

[12] Evan E Bolton, Yanli Wang, Paul A Thiessen, and Stephen H Bryant. Pubchem: integrated platform of small molecules and biological activities. In *Annual Reports in Computational Chemistry*, volume 4, pages 217–241. Elsevier, 2008. DOI: https://doi.org/10.1016/s1574-1400(08)00012-1.

[13] Hector Bombin and Miguel Angel Martin-Delgado. Topological computation without braiding. *Physical Review Letters*, 98(16):160502, 2007. DOI: https://doi.org/10.1103/physrevlett.98.160502.

[14] Earl Campbell. Shorter gate sequences for quantum computing by mixing unitaries. *Physical Review A*, 95(4):042306, 2017. DOI: https://doi.org/10.1103/physreva.95.042306.

[15] Earl Campbell. Random compiler for fast hamiltonian simulation. *Physical Review Letters*, 123(7):070503, 2019. DOI: https://doi.org/10.1103/PhysRevLett.123.070503.

[16] Earl Campbell. Early fault-tolerant simulations of the hubbard model. *Quantum Science and Technology*, 7(1):015007, 2021. DOI: https://doi.org/10.1088/2058-9565/ac3110.

[17] Yudong Cao, Jonathan Romero, Jonathan P Olson, Matthias Degroote, Peter D Johnson, Mária Kieferová, Ian D Kivlichan, Tim Menke, Borja Peropadre, Nicolas PD Sawaya, et al. Quantum chemistry in the age of quantum computing. *Chemical Reviews*, 119(19):10856–10915, 2019. DOI: https://doi.org/10.1021/acs.chemrev.8b00803.

[18] Andrew M Childs, Aaron Ostrander, and Yuan Su. Faster quantum simulation by randomization. *Quantum*, 3:182, 2019. DOI: https://doi.org/10.22331/q-2019-09-02-182.

[19] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998. DOI: https://doi.org/10.1098/rspa.1998.0164.

[20] Steven A Cuccaro, Thomas G Draper, Samuel A Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv preprint quant-ph/0410184*, 2004. DOI: https://doi.org/10.48550/arXiv.quant-ph/0410184.

[21] Alain Delgado, Pablo Antonio Moreno Casares, Roberto dos Reis, Modjtaba Shokrian Zini, Roberto Campos, Norge Cruz-Hernández, Arne-Christian Voigt, Angus Lowe, Soran Jahangiri, Miguel Angel Martin-Delgado, Jonathan E. Mueller, and Juan Miguel Arrazola. How to simulate key properties of lithium-ion batteries with a fault-tolerant quantum computer. *arXiv preprint arXiv:2204.11890*, 2022. DOI: 10.48550/ARXIV.2204.11890. URL `https://arxiv.org/abs/2204.11890`.

[22] Vincent E Elfving, Benno W Broer, Mark Webber, Jacob Gavartin, Mathew D Halls, K Patrick Lorton, and A Bochevarov. How will quantum computers provide an industrially relevant computational advantage in quantum chemistry? *arXiv preprint arXiv:2009.12472*, 2020. DOI: https://doi.org/10.48550/arXiv.2009.12472.

[23] Andrew J Ferris. Fourier transform for fermionic systems and the spectral tensor network. *Physical Review Letters*, 113(1):010401, 2014. DOI: https://doi.org/10.1103/physrevlett.113.010401.

[24] Richard P Feynman. Simulating physics with computers. In *Feynman and computation*, pages 133–153. CRC Press, 2018. DOI: https://doi.org/10.1201/9780429500459-11.

[25] Alberto Galindo and Miguel Angel Martin-Delgado. Information and computation: Classical and quantum aspects. *Reviews of Modern Physics*, 74(2):347, 2002. DOI: https://doi.org/10.1103/revmodphys.74.347.

[26] Yimin Ge, Jordi Tura, and J Ignacio Cirac. Faster ground state preparation and high-precision ground energy estimation with fewer qubits. *Journal of Mathematical Physics*, 60(2):022202, 2019. DOI: https://doi.org/10.1063/1.5027484.

[27] Craig Gidney. Halving the cost of quantum addition. *Quantum*, 2:74, 2018. DOI: https://doi.org/10.22331/q-2018-06-18-74.

[28] Joshua J Goings, Alec White, Joonho Lee, Christofer S Tautermann, Matthias Degroote, Craig Gidney, Toru Shiozaki, Ryan Babbush, and Nicholas C Rubin. Reliably assessing the electronic structure of cytochrome p450 on today's classical computers and tomorrow's quantum computers. *arXiv preprint arXiv:2202.01244*, 2022. DOI: https://doi.org/10.48550/arXiv.2202.01244.

[29] Harper R. Grimsley, S. Economou, Edwin Barnes, and Nicholas J. Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications*, 10, 2019. DOI: https://doi.org/10.1038/s41467-019-10988-2.

[30] Matthew B. Hastings, Dave Wecker, Bela Bauer, and Matthias Troyer. Improving quantum algorithms for quantum chemistry. *Quantum Information and Computation*, 15(1–2): 1–21, jan 2015. ISSN 1533-7146. DOI: https://doi.org/10.26421/qic15.1-2-1.

[31] Frank Jensen. Atomic orbital basis sets. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 3(3):273–295, 2013. DOI: https://doi.org/10.1002/wcms.1123.

[32] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. Chow, and J. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549:242–246, 2017. DOI: https://doi.org/10.1038/nature23879.

[33] Julia Kempe, Alexei Kitaev, and Oded Regev. The complexity of the local hamiltonian problem. *SIAM Journal on Computing*, 35(5):1070–1097, 2006. DOI: https://doi.org/10.1137/s0097539704445226.

[34] Mária Kieferová, Artur Scherer, and Dominic W Berry. Simulating the dynamics of time-dependent hamiltonians with a truncated dyson series. *Physical Review A*, 99(4):042314, 2019. DOI: https://doi.org/10.1103/physreva.99.042314.

[35] Isaac H Kim, Ye-Hua Liu, Sam Pallister, William Pol, Sam Roberts, and Eunseok Lee. Fault-tolerant resource estimate for quantum chemical simulations: Case study on li-ion battery electrolyte molecules. *Physical Review Research*, 4(2):023019, 2022. DOI: https://doi.org/10.1103/physrevresearch.4.023019.

[36] Ian D Kivlichan, Craig Gidney, Dominic W Berry, Nathan Wiebe, Jarrod R McClean, Wei Sun, Zhang Jiang, Nicholas C Rubin, Austin Fowler, Alán Aspuru-Guzik, et al. Improved fault-tolerant quantum simulation of condensed-phase correlated electrons via trotterization. *Quantum*, 4:296, 2020. DOI: https://doi.org/10.22331/q-2020-07-16-296.

[37] Jorge Kohanoff. *Electronic structure calculations for solids and molecules: theory and computational methods*. Cambridge university press, 2006. DOI: https://doi.org/10.1017/CBO9780511755613.

[38] Emiel Koridon, Saad Yalouz, Bruno Senjean, Francesco Buda, Thomas E O'Brien, and Lucas Visscher. Orbital transformations to reduce the 1-norm of the electronic structure hamiltonian for quantum computing applications. *Physical Review Research*, 3(3):033127, 2021. DOI: https://doi.org/10.1103/physrevresearch.3.033127.

[39] Joonho Lee, Dominic W Berry, Craig Gidney, William J Huggins, Jarrod R McClean, Nathan Wiebe, and Ryan Babbush. Even more efficient quantum computations of chemistry through tensor hypercontraction. *PRX Quantum*, 2(3):030305, 2021. DOI: https://doi.org/10.1103/prxquantum.2.030305.

[40] Zhendong Li, Junhao Li, Nikesh S Dattani, CJ Umrigar, and Garnet Kin-Lic Chan. The electronic complexity of the ground-state of the femo cofactor of nitrogenase as relevant to quantum simulations. *The Journal of Chemical Physics*, 150(2):024302, 2019. DOI: https://doi.org/10.1063/1.5063376.

[41] Lin Lin and Yu Tong. Near-optimal ground state preparation. *Quantum*, 4:372, 2020. DOI: https://doi.org/10.22331/q-2020-12-14-372.

[42] Seth Lloyd. Universal quantum simulators. *Science*, pages 1073–1078, 1996. DOI: https://doi.org/10.1126/science.273.5278.1073.

[43] Guang Hao Low and Isaac L Chuang. Optimal hamiltonian simulation by quantum signal processing. *Physical Review Letters*, 118(1):010501, 2017. DOI: https://doi.org/10.1103/physrevlett.118.010501.

[44] Guang Hao Low and Isaac L Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, 2019. DOI: https://doi.org/10.22331/q-2019-07-12-163.

[45] Guang Hao Low and Nathan Wiebe. Hamiltonian simulation in the interaction picture. *arXiv preprint arXiv:1805.00675*, 2018. DOI: https://doi.org/10.48550/arXiv.1805.00675.

[46] Guang Hao Low, Vadym Kliuchnikov, and Luke Schaeffer. Trading t-gates for dirty qubits in state preparation and unitary synthesis. *arXiv preprint arXiv:1812.00954*, 2018. DOI: https://doi.org/10.48550/arXiv.1812.00954.

[47] Sam McArdle, Tyson Jones, Suguru Endo, Y. Li, S. Benjamin, and Xiao Yuan. Variational ansatz-based quantum simulation of imaginary time evolution. *npj Quantum Information*, 5:1–6, 2018. DOI: https://doi.org/10.1038/s41534-019-0187-2.

[48] Sam McArdle, Earl Campbell, and Yuan Su. Exploiting fermion number in factorized decompositions of the electronic structure hamiltonian.

Physical Review A, 105(1):012403, 2022. DOI: https://doi.org/10.1103/physreva.105.012403.

[49] Jarrod R McClean, Nicholas C Rubin, Kevin J Sung, Ian D Kivlichan, Xavier Bonet-Monroig, Yudong Cao, Chengyu Dai, E Schuyler Fried, Craig Gidney, Brendan Gimby, et al. Openfermion: the electronic structure package for quantum computers. *Quantum Science and Technology*, 5(3):034014, 2020. DOI: https://doi.org/10.1088/2058-9565/ab8ebc.

[50] Mario Motta, Erika Ye, Jarrod R McClean, Zhendong Li, Austin J Minnich, Ryan Babbush, and Garnet Kin Chan. Low rank representations for quantum simulation of electronic structure. *npj Quantum Information*, 7(1):1–7, 2021. DOI: https://doi.org/10.1038/s41534-021-00416-z.

[51] Felix Motzoi, Michael P Kaicher, and Frank K Wilhelm. Linear and logarithmic time compositions of quantum many-body operators. *Physical Review Letters*, 119(16):160503, 2017. DOI: https://doi.org/10.1103/physrevlett.119.160503.

[52] Edgard Muñoz-Coreas and Himanshu Thapliyal. T-count optimized design of quantum integer multiplication. *arXiv preprint arXiv:1706.05113*, 2017. DOI: https://doi.org/10.48550/arXiv.1706.05113.

[53] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[54] Alberto Peruzzo, Jarrod R McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213, 2014. DOI: https://doi.org/10.1038/ncomms5213.

[55] David Poulin, Matthew B Hastings, Dave Wecker, Nathan Wiebe, Andrew C Doherty, and Matthias Troyer. The trotter step size required for accurate quantum simulation of quantum chemistry. *arXiv preprint arXiv:1406.4920*, 2014. DOI: https://doi.org/10.26421/qic15.5-6-1.

[56] David Poulin, Alexei Kitaev, Damian S Steiger, Matthew B Hastings, and Matthias Troyer. Quantum algorithm for spectral measurement with a lower gate count. *Physical Review Letters*, 121(1):010501, 2018. DOI: https://doi.org/10.1103/physrevlett.121.010501.

[57] Abhishek Rajput, Alessandro Roggero, and Nathan Wiebe. Hybridized methods for quantum simulation in the interaction picture. *arXiv preprint arXiv:2109.03308*, 2021. DOI: https://doi.org/10.48550/arXiv.2109.03308.

[58] Markus Reiher, Nathan Wiebe, Krysta M Svore, Dave Wecker, and Matthias Troyer. Elucidating reaction mechanisms on quantum computers. *Proceedings of the National Academy of Sciences*, 114(29):7555–7560, 2017. DOI: https://doi.org/10.1073/pnas.1619152114.

[59] Elvira R Sayfutyarova, Qiming Sun, Garnet Kin-Lic Chan, and Gerald Knizia. Automated construction of molecular active spaces from atomic valence orbitals. *Journal of Chemical Theory and Computation*, 13(9):4063–4078, 2017. DOI: https://doi.org/10.1021/acs.jctc.7b00128.s001.

[60] Peter Selinger. Efficient clifford+t approximation of single-qubit operators. *Quantum Info. Comput.*, 15(1–2):159–180, jan 2015. ISSN 1533-7146. DOI: https://doi.org/10.26421/qic15.1-2-10.

[61] Vivek V Shende, Stephen S Bullock, and Igor L Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006. DOI: https://doi.org/10.1109/tcad.2005.855930.

[62] Yuan Su, Dominic W Berry, Nathan Wiebe, Nicholas C Rubin, and Ryan Babbush. Fault-tolerant quantum simulations of chemistry in first quantization. *PRX Quantum*, 2(4):040332, 2021. DOI: https://doi.org/10.1103/prxquantum.2.040332.

[63] Yuan Su, Hsin-Yuan Huang, and Earl T Campbell. Nearly tight trotterization of interacting electrons. *Quantum*, 5:495, 2021. DOI: https://doi.org/10.22331/q-2021-07-05-495.

[64] Qiming Sun, Timothy C Berkelbach, Nick S Blunt, George H Booth, Sheng Guo, Zhendong Li, Junzi Liu, James D McClain, Elvira R Sayfutyarova, Sandeep Sharma, et al. Pyscf: the python-based simulations of chemistry framework. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 8(1):e1340, 2018. DOI: https://doi.org/10.1002/wcms.1340.

[65] Masuo Suzuki. Fractal decomposition of exponential operators with applications to many-body theories and monte carlo simulations. *Physics Letters A*, 146(6):319–323, 1990. DOI: https://doi.org/10.1016/0375-9601(90)90962-n.

[66] Masuo Suzuki. General theory of fractal path integrals with applications to many-body theories and statistical physics. *Journal of Mathematical Physics*, 32(2):400–407, 1991. DOI: https://doi.org/10.1063/1.529425.

[67] Himanshu Thapliyal, TSS Varun, Edgard Munoz-Coreas, Keith A Britt, and Travis S Humble. Quantum circuit designs of integer division optimizing t-count and t-depth. In *2017 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*, pages 123–128. IEEE, 2017. DOI: https://doi.org/10.1109/inis.2017.34.

[68] Jack E Volder. The cordic trigonometric computing technique. *IRE Transactions on electronic computers*, (3):330–334, 1959. DOI: https://doi.org/10.1109/tec.1959.5222693.

[69] Vera von Burg, Guang Hao Low, Thomas Häner, Damian S Steiger, Markus Reiher, Martin Roetteler, and Matthias Troyer. Quantum computing enhanced computational catalysis. *Physical Review Research*, 3(3):033055, 2021. DOI: https://doi.org/10.1103/physrevresearch.3.033055.

[70] Kianna Wan, Mario Berta, and Earl Campbell. A randomized quantum algorithm for statistical phase estimation. *arXiv preprint arXiv:2110.12071*, 2021. DOI: https://doi.org/10.48550/arXiv.2110.12071.

[71] Dave Wecker, Matthew B Hastings, Nathan Wiebe, Bryan K Clark, Chetan Nayak, and Matthias Troyer. Solving strongly correlated electron models on a quantum computer. *Physical Review A*, 92(6):062318, 2015. DOI: https://doi.org/10.1103/physreva.92.062318.

[72] Steven R White. Hybrid grid/basis set discretizations of the schrödinger equation. *The Journal of Chemical Physics*, 147(24):244102, 2017. DOI: https://doi.org/10.1063/1.5007066.

[73] Steven R White and E Miles Stoudenmire. Multisliced gausslet basis sets for electronic structure. *Physical Review B*, 99(8):081110, 2019.

[74] James D Whitfield, Jacob Biamonte, and Alán Aspuru-Guzik. Simulation of electronic structure hamiltonians using quantum computers. *Molecular Physics*, 109(5):735–750, 2011. DOI: https://doi.org/10.1080/00268976.2011.552441.

[75] Nathan Wiebe and Chris Granade. Efficient bayesian phase estimation. *Physical Review Letters*, 117(1):010503, 2016. DOI: https://doi.org/10.1103/physrevlett.117.010503.

[76] Ruizhe Zhang, Guoming Wang, and Peter Johnson. Computing Ground State Properties with Early Fault-Tolerant Quantum Computers. *Quantum*, 6:761, July 2022. ISSN 2521-327X. DOI: 10.22331/q-2022-07-11-761. URL `https://doi.org/10.22331/q-2022-07-11-761`.

A qDRIFT, a random Hamiltonian trotterization approach

Using Hamiltonian simulation to estimate the energy of chemical configurations can be accomplished through different methods. We will present the main ones that can be chosen from in our software package in the following appendices. We first consider the *Trotter-Suzuki decomposition* [1, 65, 66], where the time evolution of a Hamiltonian $H = \sum_{\gamma=1}^{\Gamma} w_\gamma H_\gamma$, with H_γ being a normalized Hermitian operator and w_γ a non-negative Hamiltonian coefficient, is approximated by

$$e^{-iHt} = e^{-it\sum_\gamma w_\gamma H_\gamma} \approx \left(\prod_{\gamma=1}^{\Gamma} e^{-iw_\gamma H_\gamma t/r}\right)^r. \quad (19)$$

In the limit of $r \to \infty$ the equality is exact. Notice that H and H_γ do not need to be unitary in general, only Hermitian. In contrast, e^{-iHt} is unitary, and since the electronic Hamiltonian can be written in second quantization as a Linear Combination of Unitaries, for the estimation of the cost of this method we will in fact take H_γ to be unitary, as in the rest of the described methods. In this section, we present the qDRIFT and Random Hamiltononian methods, some of the best method that uses the Trotter-Suzuki decomposition [15]. The main idea here is to reduce the complexity of the Trotter Suzuki decomposition above by randomizing the order in which the terms $e^{-iH_\gamma t/r}$ are applied. They suggest to simulate a single unitary $e^{-i\tau H_\gamma}$ randomly from an identical distribution, where $\tau = t\lambda/r$ is fixed, $\lambda = \sum_{\gamma=1}^{\Gamma} w_\gamma$, and the probability of choosing an individual unitary is weighted by the Hamiltonian coefficient w_γ. We further define $\Lambda = \max_\gamma w_\gamma$. This markovian method is referred to as the qDRIFT approach.

The qDRIFT algorithm achieves $O(\lambda^2 t^2/\epsilon_{HS})$ gate complexity, where ϵ_{HS} is the desired precision. This scaling stems from making the zeroth and first-order expansion terms of the qDRIFT quantum channel coincide with the channel that describes the unitary evolution. In contrast, the $2k$-th order (deterministic) Trotter methods have complexity $O(\Gamma^{2+1/2k}(\Lambda t)^{1+1/2k}/\epsilon_{HS}^{1/2k})$ [15]. As a consequence, the qDRIFT algorithm proves advantageous whenever $\lambda \ll \Lambda\Gamma$, which is the case for most electronic structure Hamiltonians, as the majority of terms H_γ possess small coefficients w_γ [11]. On the other hand, qDRIFT will most likely perform worse than higher-order Trotter expansion for large evolution times.

In the following, we will present the number of T gates required for performing the unitary evolution of Eq. (19) through the qDRIFT method and a second order Trotterization method, respectively. The details of this analysis are based on the supplementary material of [15] and consider the problem of estimating the ground state energy E_0 of a Hamiltonian H using quantum phase estimation. The total number of gates n of the form $e^{-i\tau H_\gamma}$ required to estimate the energy of the ground state to an additive error δ_E using qDRIFT is given by [15]

$$n = \frac{\pi^2\lambda^2}{\epsilon_{tot}\delta_E^2}\left(\frac{1+p_f}{p_f}\right)^2, \quad (20)$$

where p_f is the failure probability inherent to the quantum phase estimation algorithm and ϵ_{tot} is the total Trotter error. Similarly, using a second-order random Trotterization, this number scales as [15]

$$n = 8\Gamma^2\frac{1}{\epsilon_{tot}}\left(\frac{\pi\Lambda}{2\delta_E}\right)^{3/2}\left(\frac{1+p_f}{p_f}\right)^{3/2}. \quad (21)$$

To arrive at the cost in terms of T-gates, we need to assess the T-gate cost of simulating a gate $e^{-i\tau H_\gamma}$ and then multiply it by n as given by Eq. (20) and Eq. (21) to give an estimate for the cost of performing qDRIFT and a second-order Trotterization approach, respectively.

The difficulty here is that H_γ will be a string of Pauli operators, so we cannot just implement the rotation in each qubit separately as it is an entangling rotation. Fortunately, we can perform each $e^{-iH_\gamma\tau}$ using Clifford gates and a single C-R_z rotation [30, 51]. This, in turn can be decomposed in two R_z gates using Lemma 5.4 from [8], and each rotation implemented with $\approx 10 + 4\log(\epsilon_{SS}^{-1})$ T-gates [60].

Finally, notice that in the notation of our article, we are taking $\delta_E = 2\epsilon_{QPE}$ and $\epsilon_{tot} = \epsilon_{HS}$. Similarly ϵ_{SS} can be determined by dividing ϵ_S (which is not taken into account in [15]), by the number of unitary Pauli rotations used, $2n$.

B Taylorization-based Hamiltonian simulation

If in the previous appendix we explored the Trotter and Trotter-like methods for Hamiltonian simulation, from now on we would like to focus on so-called post-Trotter methods that allow avoiding having polynomial complexity in the Hamiltonian simulation precision ϵ_{HS}^{-1}. We will start with a method called Taylorization [3].

B.1 Method explanation

B.1.1 'Database' algorithm

The aim of the algorithm is to implement Hamiltonian simulation for $H = \sum_{\gamma=1}^{\Gamma} w_\gamma H_\gamma$, via 'Taylorization', that is, via a Taylor series:

$$e^{-iHt/r} \approx \tilde{U}_r := \sum_{k=0}^{K}\frac{(-iHt/r)^k}{k!} = \sum_{k=0}^{K}\sum_{\gamma_1,\ldots,\gamma_k=1}^{\Gamma}\frac{(-it/r)^k}{k!} w_{\gamma_1}\ldots w_{\gamma_k} H_{\gamma_1}\ldots H_{\gamma_k}, \quad (22)$$

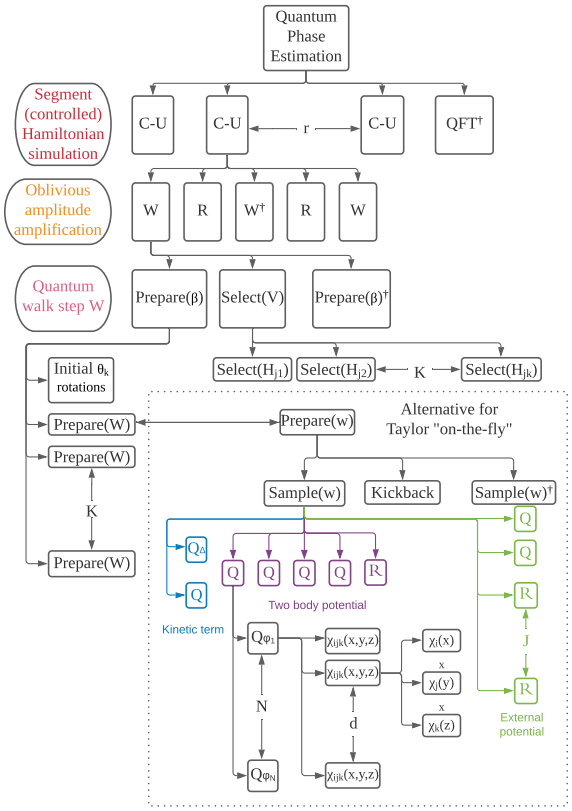

Figure 5: Abstraction level decomposition of the Taylor 'database' algorithm. The x-axis represents the time steps of the algorithm, while the y-axis is the abstraction level, higher meaning more abstract. In the lower box, we also depict the substitution one does to perform the alternative Taylor 'on-the-flight' algorithm. Notice that this does not show minor operations such as the computation of $\vec{\xi}$ or the multiplication in the last step of figure 4 from [3].

with $K = O\left(\frac{\log(r/\epsilon_{HS})}{\log\log(r/\epsilon_{HS})}\right)$. This means that in the Linear Combination of Unitaries formalism, we can write, $\tilde{U} = \sum_j \beta_j V_j$ with $\beta_j = \frac{t^k}{r^k k!} w_{\gamma_1} ... w_{\gamma_k}$ and $V_j = (-i)^k H_{\gamma_1} ... H_{\gamma_k}$.

Therefore we have to define how to implement Prepare(β) and Select(V), defined as

$$\text{Prepare}(\beta)\,|0\rangle^J = \sqrt{\frac{1}{s}} \sum_j \sqrt{\beta_j}\,|j\rangle \tag{23a}$$

depicted in figure 1 of [3], and

$$\text{Select}(V)\,|j\rangle\,|\psi\rangle = |j\rangle\,V_j\,|\psi\rangle\,. \tag{23b}$$

These operators use Prepare(W) and Select(H) respectively:

$$\text{Prepare}(W)\,|0\rangle^{\otimes\lceil\log_2\Gamma\rceil} = \sqrt{\frac{1}{\lambda}} \sum_{\gamma=1}^{\Gamma} \sqrt{w_\gamma}\,|\gamma\rangle \tag{24a}$$

with $\lambda = \sum_j |w_j| = O(N^4)$, and

$$\text{Select}(H)\,|\gamma\rangle\,|\psi\rangle = |\gamma\rangle\,H_\gamma\,|\psi\rangle\,, \tag{24b}$$

or in other words

$$\text{Select}(H)\,|ijkl\rangle\,|\psi\rangle = |ijkl\rangle\,a_i^\dagger a_j^\dagger a_k a_l\,|\psi\rangle\,. \tag{24c}$$

To implement (24c) we have to transform the creation and annihilation operators according to eq. 20 and 21 from [3]. This same article suggests introducing four additional qubits so that eq. 23 and 24 from [3] are finally used, containing only controlled Pauli operators.

Using those operators, we define the quantum walk step implementing $\tilde{U}_r$ (figure 2 in [3])

$$\mathcal{W} = (\text{Prepare}(\beta) \otimes \mathbf{1})^\dagger \text{Select}(V) (\text{Prepare}(\beta) \otimes \mathbf{1}) \tag{25a}$$

$$\mathcal{W}\,|0\rangle^J\,|\psi\rangle = \frac{1}{s}\,|0\rangle\,\tilde{U}_r\,|\psi\rangle + \sqrt{1 - \frac{1}{s^2}}\,|\Phi\rangle\,. \tag{25b}$$

To be able to use oblivious amplitude amplification, we need $s \approx 2$ [9], what can be achieved if $r = \lambda t / \ln 2$. Then $s = \sum_j |\beta_j| = \sum_{k=0}^{K} \frac{1}{k!} \ln 2^k \approx 2$.

B.1.2 'On-the-fly' algorithm

The main difference with the 'database algorithm' is that this algorithm aims to compute the integrals on-the-fly.

One starts observing that the Hamiltonian is constant in time, but at the same time it can be expressed as a spatial integral over a given region $\mathcal{Z}$, given that it decays exponentially outside it

$$H = \int_{\mathcal{Z}} \mathcal{H}(\vec{z}) d\vec{z} \approx \frac{\mathcal{V}}{\mu} \sum_{\rho=1}^{\mu} \mathcal{H}(\vec{z}). \tag{26}$$

As done in previous appendices, we divide the Hamiltonian evolution into segments U_r,

$$U_r \approx \sum_{k=0}^{K} \frac{(-it/r)^k}{k!} \int_{\mathcal{Z}} \mathcal{H}(\vec{z}_1)...\mathcal{H}(\vec{z}_k) d\vec{\boldsymbol{z}}. \tag{27}$$

If we substitute the integrals by Riemannian sums, $\mathcal{H}(\vec{z}) = \sum_{\gamma=1}^{\Gamma} w_\gamma(\vec{z}) H_\gamma$,

$$U_r \approx \sum_{k=0}^{K} \frac{(-it\mathcal{V})^k}{r^k \mu^k k!} \cdot \cdot \sum_{\gamma_1,...,\gamma_k=1}^{\Gamma} \sum_{\rho_1,...,\rho_k=1}^{\mu} w_{\gamma_1}(\vec{z}_{\rho_1})...w_{\gamma_k}(\vec{z}_{\rho_k}) H_{\gamma_1}...H_{\gamma_k} \tag{28}$$

Now, the question is how to prepare $w_{\gamma_i}(\vec{z}_{\rho_i})$ in the amplitudes. What the article does is first assume we have a method sample(w) such that

$$\text{sample}(w) |\gamma\rangle |\rho\rangle |0\rangle^{\otimes \lceil \log_2 M \rceil} = |\gamma\rangle |\rho\rangle |\tilde{w}_\gamma(\vec{z}_\rho)\rangle \tag{29}$$

with $\tilde{w}_\gamma(\vec{z}_\rho)$ an approximation of $w_\gamma(\vec{z}_\rho)$. Then the preparation procedure of the amplitudes consists of calculating the coefficients $w_{\gamma,m}(\vec{z}_\rho) \in \{\pm 1\}$ of a superposition such that $w_\gamma(\vec{z}) \approx \zeta \sum_{m=1}^{M} w_{\gamma,m}(\vec{z})$; $\zeta = \Theta\left(\frac{\epsilon_H}{\Gamma \mathcal{V} t}\right)$. To do that, defining $|l\rangle = |\gamma\rangle |m\rangle |\rho\rangle$, one performs *Kickback*:

$$|l\rangle |\tilde{w}_\gamma(\vec{z}_\rho)\rangle \to \begin{cases} |l\rangle |\tilde{w}_\gamma(\vec{z}_\rho)\rangle & \tilde{w}_\gamma(\vec{z}_\rho) > (2m - M)\zeta \\ i |l\rangle |\tilde{w}_\gamma(\vec{z}_\rho)\rangle & \tilde{w}_\gamma(\vec{z}_\rho) \leq (2m - M)\zeta \end{cases} \tag{30}$$

before uncomputing sample(w).

In summary, to prepare the amplitudes, one calculates sample(w) in the basis, performs (30) in a superposition of $|m\rangle$, and uncomputes the register prepared by sample(w). We will call such procedure Prepare(w):

$$\text{Prepare}(w) |0\rangle^{\otimes \lceil \log_2 L \rceil} = \sqrt{\frac{1}{\lambda'}} \sum_{l=1}^{L} \sqrt{\frac{\zeta \mathcal{V}}{\mu} w_{\gamma,m}(\vec{z}_\rho)} |l\rangle, \tag{31}$$

where $\lambda' = L\frac{\zeta \mathcal{V}}{\mu} = \Theta(\Gamma \mathcal{V} \max_{\vec{z},\gamma} |w_\gamma(\vec{z})|)$; $L = \Theta(\Gamma \mu M)$ and $M = \Theta(\max_{\vec{z},\gamma} |w_\gamma(\vec{z})|/\zeta)$. Additionally, due to equation 66 from [3] we know that

$$\mathcal{V} \max_{\vec{z},\gamma}(|w_\gamma(\vec{z})|) = 2^6 \varphi_{\max}^4 x_{\max}^5, \tag{32}$$

where the 2^6 is due to there being a hypercube with $(2x_{\max}/\delta x)^6$ terms.

This means that this alternative algorithm is similar to the 'database' one, but substitutes Prepare(W) with Prepare(w) that we just explained. The preparation over $|k\rangle$ is similar to the one depicted in figure 1 of [3], except that λ gets substituted by λ'.

The final, important detail we have to explain is how to perform the sample(w) routine. We want to calculate

$$\begin{aligned} w_\gamma(\vec{z}) = h_{ijkl}(\vec{x}, \vec{y}) &= \frac{\varphi_i^\dagger(\vec{x}) \varphi_j^\dagger(\vec{y}) \varphi_l(\vec{x}) \varphi_k(\vec{y})}{|\vec{x} - \vec{y}|} \\ &= \varphi_i^\dagger(\vec{x}) \varphi_j^\dagger(\vec{x} - \vec{\xi}) \varphi_l(\vec{x}) \varphi_k(\vec{x} - \vec{\xi}) |\vec{\xi}| \sin(\theta), \end{aligned} \tag{33a}$$

with $\vec{\xi} = \vec{x} - \vec{y}$ and θ the polar angle of $\vec{\xi}$; as well as

$$\begin{aligned} w_\gamma(\vec{z}) &= h_{ik}(\vec{x}) \\ &= \varphi_i^\dagger(\vec{x}) \left(-\sum_{j=0,1,2} \frac{\nabla_j^2}{2} - \sum_{j=0,...,J} \frac{Z_j}{|\vec{R}_j - \vec{x}|} \right) \varphi_k(\vec{x}) \\ &= -\varphi_i^\dagger(\vec{x}) \frac{\nabla^2}{2} \varphi_k(\vec{x}) \\ &\quad - \sum_j Z_j |\vec{\xi}_j| \sin(\theta_j) \varphi_i^\dagger(\vec{R}_j - \vec{\xi}_j) \varphi_k(\vec{R}_j - \vec{\xi}_j) \end{aligned} \tag{33b}$$

again transforming to polar coordinates in the external potential, $\vec{\xi}_j = \vec{R}_j - \vec{x}$. We need a subroutine Q to calculate the integrals.

$$\begin{aligned} Q &= \prod_{j=1}^{N} |j\rangle \langle j| \otimes Q_{\varphi_j}, \\ Q_{\varphi_j} |\rho\rangle |0\rangle^{\otimes \lceil \log_2 M \rceil} &= |\rho\rangle |\varphi_j(\vec{z}_\rho)\rangle. \end{aligned} \tag{34}$$

From the previous equation, one can see that the complexity of Q is N times the complexity of Q_{φ_j}. Notice that we will have to integrate over the space volume $\mathcal{V}$, summing over its discretization.

B.2 How to compute its cost

B.2.1 'Database' algorithm

We will use figure 5 as the main guide to compute the cost of the different abstraction levels. The first thing we have to take is the simulation time required, fixed by the error in the Phase Estimation algorithm, ϵ_{QPE}. One takes the number of segments $\tilde{U}_r$ to be

$$r = \frac{\lambda t}{\ln 2} = \frac{\pi \lambda}{\epsilon_{QPE} \ln 2}. \tag{35}$$

Another important parameter is the value of K, that controls the number of Prepare(W) in Prepare(β) and Select(H) in Select(V), which we can take from [45] to be

$$K = \left\lceil -1 + \frac{2 \log(2r/\epsilon_{HS})}{\log(\log(2r/\epsilon_{HS}) + 1)} \right\rceil. \tag{36}$$

The final aspects to take into account are:

1. θ_k **initial rotations**. This can be done using $K-1$ controlled R_y rotations.

2. **Prepare(W)** The cost of an arbitrary state preparation for can be estimated as $2^{\lceil \log_2 N^4 \rceil + 1}$ arbitrary rotations, using the protocol from [61], as it is preferable to encode $|ijkl\rangle$ instead of a continuous register that later on gets converted to that. This will be the most expensive part of the algorithm.

3. **Select(H)** First we have to specify how to create the circuit for each operator $a_{j,q}$ (analogously $a^\dagger_{j,q}$). For that we iterate over $n \in \{1, ..., N\}$. If $j = n$ we apply a σ_x or $\pm i\sigma_y$ as dictated by $|q\rangle$, if $j < n$ then we apply σ_z.

   The equality case can be performed via multi-controller Pauli operators. For each creation/annihilation operator, there will be $4N$ options due to the possible values of $|j\rangle\,|q\rangle$. We have to control on one qubit of register $|k\rangle$ encoded in unary to take into account the amplitude term corresponding to $\frac{(t/r)^k}{k!}$, on $|j\rangle$ with $\lceil \log_2 N \rceil$ qubits, and on $|q\rangle$; we will need to resort to multi controlled gate decomposition.

   To avoid the comparison in the case of $n < j$ we can create an accumulator. That is, when $n = j$ we switch an ancilla from $|1\rangle \to |0\rangle$, and controlled on such ancilla (and the unary register $|k\rangle$), at each step we perform σ_z on the $n-$th register of $|\psi\rangle$. This means N Toffolis and N multi-controlled (on $\lceil \log_2 N \rceil$ qubits) Not gates due to the equality comparison.

B.2.2 'On-the-fly' algorithm

To compute the cost of the 'on-the-fly' variation of this algorithm, the key step is substituting the Prepare(W) operator by something less expensive. The way we do this is by computing the one and two body integrals on the fly, by creating a sign-weighted superposition in register $|\rho\rangle$. Such superposition will use $\lceil \log_2 \mu \rceil$ qubits and can take values from 0 to $\mu - 1$ where

$$\begin{aligned}\mu &\approx \left(\frac{2r \times 6K}{\epsilon_H}(4\varphi'_{\max} + \varphi_{\max}/x_{\max})\varphi^3_{\max}x^6_{\max}\right)^6 \\ &= \Theta\left(\left(\frac{N^4 t}{\epsilon_H}(\varphi'_{\max} + \varphi_{\max}/x_{\max})\varphi^3_{\max}x^6_{\max}\right)^6\right)\end{aligned} \tag{37}$$

as can be seen from equations 73 and 74, and the text in the paragraph before equation 61, from [3]. Although this is a large number, it will only appear logarithmically in the number of qubits in the $|\rho\rangle$ register as explained in (28), so does not represent a too large complexity overhead. Notice that from equation 60 in [3], $r = \frac{\lambda' t}{\ln 2} = \frac{t}{\ln 2}\Gamma\mathcal{V}\max_{\vec{z},\gamma}(|w_\gamma(\vec{z})|)$, and the factor of 4 in front of $\varphi'_{\max}$ appear because we were deriving $\varphi^4_{\max}$; whereas the 2 appears because if we assume a hypercube, there should be $(2x_{\max}/\delta x_{\max})^6$ blocks in the discretization. Additionally, we can choose the coordinate system centered around the orbital such that $x_{\max} = O(\log(Nt/\epsilon_H)) = C\log(Nt/\epsilon_H)$, C a constant given by the software package users. $\varphi_{\max}$ will not depend on N. Similarly, since ζ is ϵ_H divided by the number of integral terms calculated in the process,

$$M = \frac{\max_{\vec{z},\gamma}(|w_\gamma(\vec{z})|)}{\zeta} = \frac{6Kr\Gamma\mathcal{V}\max_{\vec{z},\gamma}(|w_\gamma(\vec{z})|)}{\epsilon_H}, \tag{38}$$

where we can use the expressions from (32).

The final contribution we should take into account is that of the arithmetic operations required to calculate $\varphi_j(\vec{z}_\rho)$, which will also depend on the basis function we are using.

For that we will be using quantum addition [27], multiplication [52] and integer division [67]. The respective T-gate costs are $4n + O(1)$, $21n^2 - 14$ and $14n^2 + 7n + 7$, where n is the number of digits, $n = \lceil \log_2 \mu \rceil/3$, as there are three coordinates. Additionally, performing comparison between two numbers [20] can be done using $2n$ Toffoli gates if each of the inputs to compare is length n, so $8n$ T-gates.

To calculate the number of operations needed, we have to first remember that we are using a Gaussian basis set. In such basis, we expand the wave function as $\phi = \sum_{i=1}^M c_i\chi_i$. Each $\chi_j(x, y, z) = (x - X)^k(y - Y)^l(z - Z)^m e^{-\zeta_i(\mathbf{r}-\mathbf{R})^2}$, where (X, Y, Z) indicate the center of the atom, and $k + l + m$ is the angular momentum (eg. $k + l + m = 1$ means p-type basis etc. We assume that we only use up to d basis). The orbitals are usually contracted $\kappa_j = \sum_{i=1}^d d_{ij}\chi_i$ and $\phi = \sum_{j=1}^N c_j\kappa_j$. Each κ_j is one of the N basis functions that we use. More information on the topic of Gaussian basis sets might be found in a recent review [31].

In any case, to calculate each basis function $\kappa_j = \varphi_j$ we have to do the following:

1. Calculate $(x - X)$, $(y - Y)$, and $(z - Z)$, using $12n + O(1)$ T gates.

2. Calculate $(\mathbf{r}-\mathbf{R})^2 = (x-X)^2+(y-Y)^2+(z-Z)^2$, with cost $3(21n^2 - 14)$ for the multiplications, that is the leading cost. The sums mean $8n+O(1)$ additional cost.

3. Calculate the exponential $\zeta_i(\mathbf{r}-\mathbf{R})^2$ with a single multiplication, at T-gate cost $(21n^2 - 14)$.

4. $e^{-\zeta_i(\mathbf{r}-\mathbf{R})^2}$ via a Taylor series. Expanding to order o means $o - 1$ multiplications and divisions, and o sums.

5. The error in the previous expansion can be bounded as $\max\left(\zeta_i(\mathbf{r} - \mathbf{R})^2\right)^o/o!$

6. To construct $\chi_j(x, y, z)$ we need 3 multiplications, so the cost is $\approx 3(21n^2 - 14)$.

7. Each κ_j will be a sum of weighted exponentials, so the previous cost should be multiplied by d, the number of terms in such sum.

The number of terms d in each κ_j depends on the basis used, but it can be seen in tables 1-4 from [31] that the number of primitive basis sets χ_i that form each κ_j does not exceed 6 functions in the case of segmented basis sets (sparse d_{ij}), so we will take $d = 6$. However, if the basis set is general-contracted, d_{ij} is dense and the number might be much greater.

Once we have computed $\kappa_j = \varphi_j(\vec{x})$, we want to compute $\tilde{w}_\gamma(\vec{z})$:

- Whenever we have to compute $\vec{\xi}_j$ or $\vec{\xi}_j$, the cost is $12n + O(1)$ T-gates.

- Performing $\mathcal{R}|\vec{\xi}\rangle|0\rangle \mapsto |\vec{\xi}\rangle||\vec{\xi}|\sin\theta\rangle$, and similarly for $\vec{\xi}_j$. To do that, observe that $|\vec{\xi}|\sin\theta = \sqrt{\vec{x}_x^2 + \vec{x}_y^2}$, so we need two multiplications at cost $2(21n^2 - 14)$, one sum at T-gate cost $4n + O(1)$, and a square root calculation. We compute the square root using the Babylonian method, which only involves a sum and a division per order.

- $\nabla^2\chi_k(x) = \left(4x^2 - 2 + 4k - (1+k)/x^2\right)\chi_k(x)$. If we call the parenthesis $a_k(x)$, then $\nabla^2\chi_{ijk}(x,y,z) = (a_i(x) + a_j(y) + a_k(z))\chi_{ijk}$. Computing $a_i(x)$ can be done using 4 sums, 1 multiplication (x^2 term) and 1 division. This is because multiplying by 4 is free, just shifting bit positions. This has to be multiplied by 3 to take into account the three coordinates in the Laplacian, and done before the combination of the d functions into a single $\kappa_j = \varphi_j$.

In a similar fashion can Q_Δ be computed, for the sake of a name for outputting $\nabla^2\varphi$.

Overall, the cost of Sample(w) is

- Two-body term: 4Q $+\mathcal{R}$ + 4 multiplication + computation of $\vec{\xi}$.

- Kinetic term: $Q + Q_\Delta$ + multiplication.

- External potential term: 2Q $+J \times \mathcal{R}$ + J multiplications by Z_j and $J-1$ sums + J computations of $\vec{\xi}_j$.

Remember that in the previous calculations we are taking $n = \lceil \log_2 \mu \rceil/3$.

The cost of the rotation *Kickback* between the two applications of Sample(w) can be seen as a controlled rotation on the result of a comparison with $\lceil \log_2 \mu \rceil$ qubits. This requires one sum, one multiplication, and one comparison, which should be done twice to uncompute the result once the rotation has happened. From the previous, the cost of the 'on-the-fly' version of algorithm [3] can be computed using figure 5.

B.3 How to adapt the Hamiltonian simulation to control the direction of the time evolution

Quantum Phase Estimation requires being able to control the time direction of the Hamiltonian evolution of a segment. We do that by slightly modifying the Select(V) operator: if we want to simulate $e^{-iHt/r}$, for $k = 4j + 1$ we apply a C-S† operation (to apply $-i$ phase) and C-S if $k = 4j + 3$, while if we instead want to simulate $e^{iHt/r}$ additionally apply C-X in those situations to flip the sign. Here the Control bits are the value of k and the control qubits in Quantum Phase Estimation.

Adapting the Hamiltonian simulation method for Phase Estimation operation then amounts to two multi-controlled Not gates, with $K/2 + 1$ controls because k is encoded in unary and we are using Bayesian Phase Estimation with a single control ancilla.

C Configuration interaction and first quantization

C.1 Method explanation

In the previous section, we saw how to use Taylorization as a Hamiltonian simulation method in second quantization. Here, we explain the approach of [4], which relies on the same approach but in first quantization, in a formulation called Configuration Interaction. The general structure of the algorithm will consequently be similar.

In the Configuration Interaction representation one writes $|\alpha\rangle = |\alpha_0, ..., \alpha_{\eta-1}\rangle$, where each α_i indicates an occupied orbital. The determinant of the corresponding wave functions is an antisymmetric function called Slater determinant and represents the state of the system

$$\langle \vec{r}_0, ...\vec{r}_{\eta-1}|\alpha\rangle = \frac{1}{\sqrt{\eta!}}\left|\begin{pmatrix} \varphi_{\alpha_0}(\vec{r}_0) & \cdots & \varphi_{\alpha_{\eta-1}}(\vec{r}_0) \\ \vdots & & \vdots \\ \varphi_{\alpha_0}(\vec{r}_{\eta-1}) & \cdots & \varphi_{\alpha_{\eta-1}}(\vec{r}_{\eta-1}) \end{pmatrix}\right|. \tag{39}$$

An important aspect of this method is that it can only be applied with local basis functions, such as Gaussian orbitals, but not the plane-wave basis. The reason is that at one point one has to bound the error by approximating Hamiltonian integrals from Riemannian sums, and bounding the error is only possible if we are restricted to a local volume of space. To make it work with molecular orbitals appearing in the Hartree-Fock procedure, one can use the operator $U = \exp\left(-\sum_{ij}\kappa_{ij}a_i^\dagger a_j\right)$ that changes the basis and may be applied using $\tilde{O}(N^2)$ gates [71]. κ here is an antihermitian matrix that is obtained by the self-consistent Hartree Fock procedure.

Expressing the Configuration Interaction Hamiltonian as a linear combination of unitaries is not efficient. On the other hand, though, it can be expressed as a sparse matrix, called Configuration Interaction (CI), whose elements are a sum of integrals.

The Slater-Condon rules indicate how to compute those matrix elements, based on one- and two-body integrals [4]. Because of them, the sparsity of the Configuration Interaction matrix is

$$\begin{aligned} d &= \binom{\eta}{2}\binom{N-\eta}{2} + \binom{\eta}{1}\binom{N-\eta}{1} + 1 \\ &= \frac{\eta^4}{4} - \frac{\eta^3 N}{2} + \frac{\eta^2 N^2}{2} + O(\eta^2 N + \eta N^2) \in O(\eta^2 N^2). \end{aligned} \tag{40}$$

After decomposing the Configuration Interaction matrix in 1-sparse operators, we approximate its integrals as a Riemannian sum of self inverse operators. Finally, we construct Select($\mathcal{H}$), that applies such self inverse operators

$$\text{Select}(\mathcal{H}) \left|l\right\rangle \left|\rho\right\rangle \left|\psi\right\rangle = \left|l\right\rangle \left|\rho\right\rangle \mathcal{H}_{l,\rho} \left|\psi\right\rangle \tag{41}$$

and allows to evolve the system under the Hamiltonian. The steps are the following:

1. **Decompose the Hamiltonian into 1-sparse operators**. Such operators will be indexed by 2 4-tuples (a_1, b_1, i, p) and (a_2, b_2, j, q) that denote the differing orbitals. This tuples will be used to perform the operator

$$Q^{col} : \left|\gamma\right\rangle \left|\alpha\right\rangle \left|0\right\rangle^{\eta \lceil \log_2 N \rceil} \mapsto \left|\gamma\right\rangle \left|\alpha\right\rangle \left|\beta\right\rangle, \tag{42}$$

   within the Select operator (41). The specific algorithms for this procedure can be found in appendix A of the article of reference for this appendix [4]. These procedures require, between other things, the ability to order a list of orbitals, which we explain in Algorithm 1.

2. **Decompose each 1-sparse operator into h_{ij} and h_{ijkl}**. The Slater Condon rules sometimes requires the sum over η integrals. Here we decompose the previous sum such that only at most two integrals are summed for each term. This decomposition can be seen in section 4.2 of the original article [4]. It will allow us to write the Hamiltonian as $H = \sum_\gamma H_\gamma$, with $\Gamma = \eta + \eta(\eta-1)/2 + (N-1)\eta^2 + (N-1)^2\eta(\eta-1)/2$.

3. **Discretising the integrals into Riemannian sums**.

   Each Hamiltonian term from the previous equation might be represented as $H_\gamma^{\alpha\beta} = \int \aleph_\gamma^{\alpha\beta}(\vec{z})d\vec{z}$. Since the domain of each integral might be different, we write $H_\gamma^{\alpha\beta} \approx \sum_{\rho=1}^{\mu} \aleph_{\gamma\rho}^{\alpha\beta}$. Here is where we need the requirement that the orbitals are local.

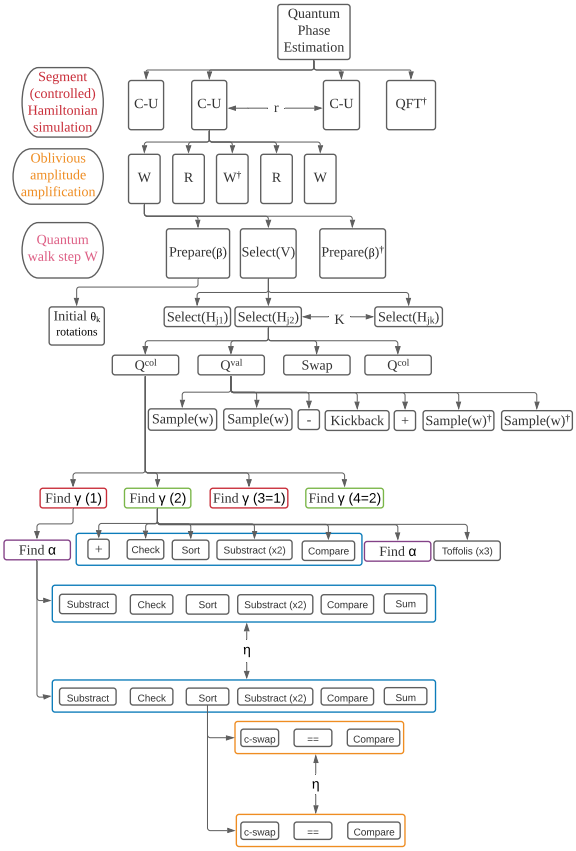

Figure 6: Abstraction level decomposition of the Configuration Interaction procedure [4]. The Sample operation shown is the same as in figure 5.

4. **Decomposition into self-inverse operators**. Finally, we decompose in a sum of $M \in \Theta(\max_{\gamma,\rho}||\aleph_{\gamma,\rho}||_{\max}/\zeta)$ self-inverse operators, using a similar strategy as in the previous section B [3]. Operators will be indexed by ρ and $l = (\gamma, m, s)$, where m controls whether a phase i is added in the Kickback, and s is sign. ρ controls the Riemmanian sum. The final decomposition can be written as $H = \zeta \sum_{l=1}^{L} \sum_{\rho=1}^{\mu} \mathcal{H}_{l,\rho}$. Using this we can perform

$$Q^{val} |l\rangle |\rho\rangle |\alpha\rangle |\beta\rangle = \mathcal{H}_{l,\rho}^{\alpha\beta} |l\rangle |\rho\rangle |\alpha\rangle |\beta\rangle , \quad (43)$$

which also appears in the Select operator.

In conclusion, one time segment of the Taylorized Hamiltonian evolution will be

$$U_r \approx \sum_{k=0}^{K} \frac{(-it\zeta)^k}{r^k k!} \sum_{l_1,\dots,l_k=0}^{L} \sum_{\rho_1,\dots,\rho_k=0}^{\mu} \mathcal{H}_{l_1,\rho_1} \dots \mathcal{H}_{l_k,\rho_k}, \quad (44)$$

where $|l\rangle = |\gamma, m, s\rangle$. The role of Prepare will be restricted to the preparation of θ angles for $\frac{(-it\zeta)^k}{r^k k!}$.

To compute the algorithm cost, we will need constants α, γ_1 and γ_2 to comply with equations 28, 29 and 30 from [4], and will bound the error from computing the Hamiltonian integrals as Riemannian sums:

- For each l there is a vector c_l such that if $||\vec{r} - \vec{c}_l|| \geq x_{\max}$ then

$$|\varphi_l(\vec{r})| \leq \varphi_{\max} \exp\left(-\frac{\alpha}{x_{\max}} ||\vec{r} - \vec{c}_l||\right) \quad (45)$$

- For each l, φ_l is twice differentiable and there exists γ_1 and γ_2 such that

$$||\nabla \varphi_l(\vec{r})|| \leq \gamma_1 \frac{\varphi_{\max}}{x_{\max}} \quad (46a)$$

and

$$||\nabla^2 \varphi_l(\vec{r})|| \leq \gamma_2 \frac{\varphi_{\max}}{x_{\max}^2} \quad (46b)$$

C.2 How to compute its cost

We will use figure 6 as a guide to computing the cost of the algorithm. There are three key differences with the cost calculated in the previous appendix. First, some parameters change. These are notably r, the number of time segments, and M, which indicates the size of register $|m\rangle$ and as a consequence influences the cost. The other two aspects that change are that we need to compute the cost of Q^{val} and Q^{col} in figure 6.

Let us start computing r, the number of segments. $r = \zeta L \mu t / \ln(2)$ (according to the paragraph before equation 68 in [4]), with $L = 2(M\Gamma)$ (the 2 because of register s in $|l\rangle = |\gamma\rangle |m\rangle |s\rangle$). The product $\mu \max_{\gamma,\rho} ||\aleph_{\rho,\gamma}|| = \mu M \zeta$ can optimized from Lemmas 1-3 in the original article [4], so

$$r = 2\Gamma t (\mu M \zeta) / \ln(2), \quad (47)$$

with $t = \pi/\epsilon_{QPE}$ and

$$\Gamma = \binom{\eta}{2}\binom{N-\eta}{2} + \binom{\eta}{1}\binom{N-\eta}{1} + 1 = \frac{\eta^4}{4} - \frac{\eta^3 N}{2} + \frac{\eta^2 N^2}{2} + O(\eta^2 N + \eta N^2) \in O(\eta^2 N^2). \quad (48)$$

To compute M, similarly as in the previous appendix

$$M = \Theta\left(\frac{\max_{\gamma,\rho} ||\aleph_{\rho,\gamma}||}{\zeta}\right), \quad (49)$$

and in the previous appendix we saw that ζ is the error that we allow, modelled as the error budget for this error source ϵ_H, divided by the number of times we called the decomposition, $\Gamma \mathcal{V} r$. The reason why $\mathcal{V}$ appeared in place of μ is because instead of writing

$$H_\gamma = \sum_\rho w_\gamma(\vec{z}_\rho) \quad (50a)$$

we were taking

$$H_\gamma = \frac{\mathcal{V}}{\mu} \sum_\rho w_\gamma(\vec{z}_\rho), \quad (50b)$$

so the precision must be scaled correspondingly. In this case however,

$$H_\gamma = \sum_\rho \aleph_\gamma(\vec{z}_\rho), \quad (51)$$

integrating the cell volume as a multiplicative constant in $\aleph_\gamma(\vec{z}_\rho)$, so the error has to be appropriately scaled by $\mathcal{V}/\mu$. Similarly, this time,

$$\zeta = \frac{\epsilon_H}{3 \cdot 2Kr(\#\gamma)(\#\rho)} = \frac{\epsilon_H}{6Kr\Gamma\mu}. \quad (52)$$

Since $\max_{\gamma,\rho} ||\aleph_{\rho,\gamma}||$ is bounded from Lemmas 1, 2 and 3 in [4], we can compute M. These lemmas will also depend on δ, taken to be the individual error in each of the integrals. Therefore, we should take (see paragraph before eq. 74 in [4]):

$$\delta = \frac{\epsilon_H}{6Kr}, \quad \zeta = \frac{\delta}{\Gamma\mu} \quad (53)$$

where $6K$ is the number of times these integrals are used in each segment, indicated figure 6. We can see that δ depends on r, which depends on $\mu M \zeta$, which from the previously mentioned lemmas depends on δ. We solve this by computing r such that μ times equations 39, 43 and 47 in [4] become approximate equalities to $\mu M \zeta$. This way we obtain a close result to if we had used $\delta = \epsilon_H/(6K\Gamma t)$.

Now let us turn to two main operators involved in the algorithm, Q^{val} in (43) and Q^{col} in (42). To compute the cost of Q^{val} the procedure is the same as we did in the previous appendix B. In this case, however, we will have to compute up to 2 basis functions. To do so we iterate over the different possibilities of γ to decompose in h_{ij} and h_{ijkl}.

a. $p = 0 = q$. This point requires calculating η terms of type $h_{\chi_i\chi_i}$, and $\eta(\eta-1)/2$ terms $(h_{\chi_i\chi_j\chi_i\chi_j} - h_{\chi_i\chi_j\chi_j\chi_i})$.

b. $p = 0$, $q \neq 0$. In this case there are $(N-1)\eta(\eta-1)$. terms of the form $h_{k\chi_i l\chi_i} - h_{k\chi_i\chi_i l}$, and $(N-1)\eta$ for the terms of the form h_{kl}.

c. $p \neq 0$, $q = 0$. No integrals are needed.

d. $p \neq 0$, $q \neq 0$. All of the integrals in this last point are of the form $h_{ijkl} - h_{ijlk}$. There are $(N-1)^2\eta(\eta-1)/2$ of them.

From this and the previous appendix B, the cost of Q^{val} can be readily calculated.

Computing Q^{col} requires implementing the procedure 'Find Alphas' and a more general one indicated in cases 2 and 4 in appendix A, that we will call 'Find Gammas' [4]. Both 'Find Alphas' and 'Find Gammas' require a sorting algorithm that has the peculiarity that only up to one item might be out of order, and we know its position. For that reason, we have described a possible sorting algorithm 1. To compute the cost, one should also make use of the basic operations described in table 2.

Algorithm1 Algorithm to order the orbitals $|\tilde{\alpha}\rangle$ generated from $|\beta\rangle$, shift $|p\rangle$ and position $|j\rangle$

procedure ORDER($|\beta\rangle\,|p\rangle\,|j\rangle$)
  Calculate unordered $|\tilde{\alpha}\rangle_1$ subtracting $|p\rangle$ from $|\beta_j\rangle$.
  Use Cnots to create two 'basis' copies of $|\tilde{\alpha}\rangle_1$, called $|\tilde{\alpha}\rangle_1$ and $|\tilde{\alpha}\rangle_2$
  for $i \in$ reversed(range(j)) **do**
    if then $|\tilde{\alpha}_i\rangle_1 == |\tilde{\alpha}_{i+1}\rangle_1$ then
      return Invalid ▷ If this is activated, reverse the entire computation. Thus cost $\times 2$.
    $|0\rangle_a \leftarrow (|\tilde{\alpha}_i\rangle_1 > |\tilde{\alpha}_j\rangle_1)$
    Controlled on $|\cdot\rangle_a$ swap $|\tilde{\alpha}_i\rangle_2$ and $|\tilde{\alpha}_{i+1}\rangle_2$
    Uncompute $|\cdot\rangle_a$
  Uncompute $|\tilde{\alpha}\rangle_1$
  return $|\beta\rangle\,|p\rangle\,|j\rangle\,|\tilde{\alpha}\rangle_2$

Using this and figure 6 it is relatively straightforward to compute the cost of the present algorithm. Notice however that the initial Hartree-Fock rotation $U = \exp\left(-\sum_{ij}\kappa_{kj}a_i^\dagger a_j\right)$ has not yet been implemented in the cost estimation, but it is not a dominant factor.

C.3 How to adapt the Hamiltonian simulation to control the direction of the time evolution

Adapting the Hamiltonian simulation for its use in Quantum Phase Estimation can be done as in appendix B.3. The cost can be therefore calculated in the same way.

D Introducing the QROM

D.1 Method explanation

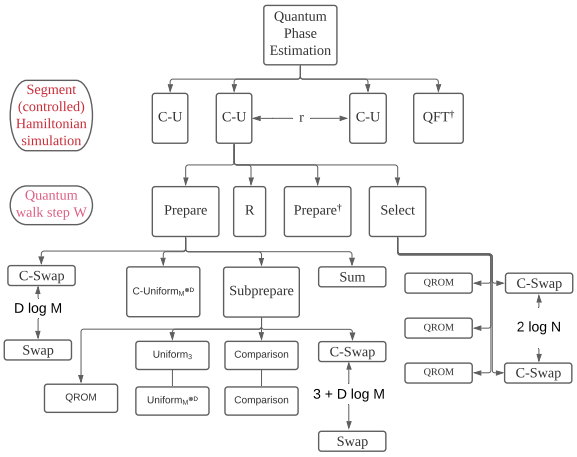

Figure 7: Abstraction level decomposition of the procedure [5].

One of the key innovations used in this method is that if instead of simulating $\mathcal{W}(H) = e^{\pm iH\tau}$ one chooses $\mathcal{W}(H) = e^{\pm i\arccos(H/\lambda)}$, one can eliminate the Taylor series error completely [5], as we already explained in section 3.3. This idea had been previously introduced [10, 56], and has the consequence that instead of phase estimating the ground state energy E_0 one phase estimates $\arccos(E_0)$. We can simulate $\mathcal{W}(H)$ with the standard quantum walk $(\text{Prepare}^\dagger \otimes \mathbf{1})\text{Select}(\text{Prepare}^\dagger \otimes \mathbf{1})$. Notice that in contrast to [62] we are using arccos instead of arcsin because, since $\arccos\theta + \arcsin\theta = \pi/2$, the change amounts to a global phase and sign change, and we want to use similar notation everywhere.

Therefore, in this appendix, we aim to explain the implementations of the Prepare and Select operators, and the key innovation of article [5], the proposal of an efficient QROM that will play an important role in both Prepare and Select. We will start with the latter. The role of the QROM is to iterate over all possible inputs preparing the corresponding outputs.

How can we construct such a unary iterator? The easiest way is just to use as control all the index qubits for each of the L values the indices can take. But this is clearly wasteful since we are often repeating the same controls over consecutive values in the indices. Therefore, [5] proposes using auxiliary qubits to hierarchically save the combinations of controls, giving rise to circuits similar to their figure 5, called the "sawtooth" circuit. This circuit, in contrast to the original, can be simplified avoiding the wasteful repetition of AND gates that we indicated previously. As shown in their figure 6, allows for converting their figure 5 to their figure 7, requiring only $(L-1)$ AND

gates. Since each AND can be constructed from 4 T gates, the unary iterator requires $4L-4$ T gates.

A variation over the previous iterator is the accumulator. Instead of directly applying the chosen gates to the target qubits, one defines an accumulator qubit, which is at state $|0\rangle$ until we control on the selected value of the indices and stays $|1\rangle$ until the end of the iterator, at which point it can be uncomputed since at the end the accumulator will be at disentangled state $|1\rangle$. A picture of this variant is figure 8 in [5]. This accumulator is specially useful because it will allow us to apply the Majorana fermion operator $|l\rangle|\psi\rangle \to |l\rangle\left(\frac{a_l^\dagger - a_l}{i}|\psi\rangle\right) = |l\rangle Y_l Z_{l-1}...Z_0|\psi\rangle$, as can be seen in figure 9 in [5].

This QROM is useful to perform the Prepare circuit. However, we will not prepare

$$|0\rangle^{\lceil \log_2 \Gamma\rceil} \mapsto \sum_{\gamma=0}^{\Gamma-1}\sqrt{\frac{w_\gamma}{\lambda}}, \tag{54}$$

but rather

$$|\mathcal{L}\rangle = \sum_{\gamma=0}^{\Gamma-1}\sqrt{\frac{w_\gamma}{\lambda}}|\gamma\rangle|\text{temp}_\gamma\rangle, \tag{55}$$

with $|\text{temp}_\gamma\rangle$ a junk register entangled with $|\gamma\rangle$. The way to ensure that this entanglement does not interfere with other computations is to ensure that the same qubits are fed into the uncomputation and that the reflection $\mathcal{R}_\mathcal{L} = (2|\mathcal{L}\rangle\langle\mathcal{L}| - \mathbf{1})$ that appears in the quantum walk step $\mathcal{W} = \mathcal{R}_\mathcal{L}\cdot\text{Select}$ is done only over state $|0\rangle$. Here, we will be looking for an algorithm that performs the following transformation:

$$|0\rangle^{\otimes(1+2\mu+2\lceil\log_2\Gamma\rceil)} \to \sum_{\gamma}^{\Gamma-1}\sqrt{\tilde{\rho}_l}|\gamma\rangle|\text{temp}_\gamma\rangle, \tag{56}$$

with $\tilde{\rho}_\gamma$ a μ-bits binary approximation to w_γ/λ. For this, one chooses

$$\mu = \left\lceil \log_2\left(\frac{2\sqrt{2}\lambda}{\Delta E}\right) + \log_2\left(1+\frac{\Delta E^2}{8\lambda^2}\right) - \log_2\left(1-\frac{||H||}{\lambda}\right)\right\rceil. \tag{57}$$

as given in equation 36 from [5]. Since the Hamiltonian is frustrated, the quotient in the last logarithm is upper bounded away from 1, and thus the last term is $O(1)$. Similarly, since $\Delta E < \lambda$, the second term can be upper bounded by $\log_2(1+1/8)$.

We will prepare this new $|\mathcal{L}\rangle$ indirectly, using a circuit that they depicted in figure 11 and called Subprepare. We start from the uniform superposition $\sum|\gamma\rangle$ and have two registers that depend on γ, $|\text{keep}_\gamma\rangle$ and $|\text{alt}_\gamma\rangle$. $|\text{keep}_\gamma\rangle$ will dictate the probability that we coherently exchange $|\gamma\rangle$ and $|\text{alt}_\gamma\rangle$. The objective is to find keep_γ and alt_γ such that in the end, we obtain the correct amplitudes. The details of the procedure can be found in section 3D in the main reference for this appendix, and it is the inverse procedure of the depicted one in their figure 13 [5].

The Hamiltonian basis explored in this technique is plane waves, with the same structure that we saw in eq. (67) [5]. The article suggests that to make the basis set as compact as possible, one may choose Gausslet basis sets, that combine some of the features of plane waves and of Gaussian waves [72, 73]. They represent however a very complex basis set, so for the time being we have not implemented it yet, working in dual waves instead.

The following question we need to answer is how to index the terms of the Hamiltonian. We will have registers $|p\rangle$ and $|q\rangle$ which in binary encode the orbitals without taking into account the spin, while $|\alpha\rangle$, and $|\beta\rangle$ will take that into account. Thus, $|p\rangle$ and $|q\rangle$ will encode numbers from 0 to $N/2-1$ (N the number of spin-orbitals) and will need $\lceil\log_2 N\rceil - 1$ qubits each. Then we will have two one-qubit registers $|U\rangle$ and $|V\rangle$, that will decide what term in the Hamiltonian to apply. Finally $|\theta\rangle$ will be used to apply a phase $(-1)^\theta$. Overall, we have the following Select operator

$$\begin{aligned}&\text{Select}\,|\theta,U,V,p,\alpha,q,\beta\rangle|\psi\rangle = \\ &(-1)^\theta|\theta,U,V,p,\alpha,q,\beta\rangle \\ &\otimes\begin{cases} Z_{p,\alpha} & U\wedge\neg V\wedge((p,\alpha)=(q,\beta)) \\ Z_{p,\alpha}Z_{q,\beta} & \neg U\wedge V\wedge((p,\alpha)\neq(q,\beta)) \\ X_{p,\alpha}\vec{Z}X_{q,\alpha} & \neg U\wedge\neg V\wedge(p<q)\wedge(\alpha=\beta) \\ Y_{p,\alpha}\vec{Z}Y_{q,\alpha} & \neg U\wedge\neg V\wedge(p>q)\wedge(\alpha=\beta) \\ \text{Undefined} & \text{Otherwise}\end{cases}.\end{aligned} \tag{58}$$

As an aside notice that p and q are three dimensional vectors whose elements take integer values in the range $[0,(N/2)^{1/3}-1]$, so we need to map (p,σ) to an integer index representing a qubit. The mapping is, for a D dimensional system ($D=3$)

$$M = (N/2)^{1/D}, \qquad f(p,\sigma) = \delta_{\sigma,\uparrow}M^D + \sum_{j=0}^{D-1}p_j M^j. \tag{59}$$

Similarly, the Prepare operator performs

$$\begin{aligned}&\text{Prepare}: |0\rangle^{\otimes(3+2\lceil\log_2 N\rceil)} \mapsto \\ &\sum_{p,\sigma}\tilde{U}(p)|\theta_p\rangle|1\rangle_U|0\rangle_V|p,\sigma,p,\sigma\rangle \\ &+\sum_{p\neq q,\sigma}\tilde{T}(p-q)|\theta^{(0)}_{p-q}\rangle|0\rangle_U|0\rangle_V|p,\sigma,q,\sigma\rangle \\ &+\sum_{(p,\alpha)\neq(q,\beta)}\tilde{V}(p-q)|\theta^{(1)}_{p-q}\rangle|0\rangle_U|0\rangle_V|p,\sigma,q,\sigma\rangle,\end{aligned} \tag{60}$$

with coefficients

$$\tilde{U}(p) = \sqrt{\frac{|T(0) + U(p) + \sum_q V(p-q)|}{2\lambda}}$$
$$\tilde{T}(p) = \sqrt{\frac{|T(p)|}{\lambda}}; \qquad \tilde{V}(p) = \sqrt{\frac{|V(p)|}{4\lambda}} \tag{61}$$

and

$$\theta_p = \frac{1 - sign(-T(0) - U(p) - \sum_q V(p-q))}{2}$$
$$\theta_p^{(0)} = \frac{1 - sign(T(p))}{2}; \qquad \theta_p^{(1)} = \frac{1 - sign(V(p))}{2}. \tag{62}$$

To implement Prepare, first, we prepare a unitary operator called Subprepare, which acts as

$$|0\rangle^{\otimes(2+\log_2 N)} \mapsto$$
$$\sum_{d=0}^{N-1} \Big(\tilde{U}(d)\,|\theta_d\rangle\,|1\rangle_U\,|0\rangle_T + \tilde{T}(d)\,|\theta_d^{(0)}\rangle\,|0\rangle_U\,|0\rangle_V \tag{63}$$
$$+ \tilde{V}(d)\,|\theta_d^{(1)}\rangle\,|0\rangle_U\,|1\rangle_V\Big)\,|d\rangle\,.$$

The construction of Select, Subprepare and Prepare can be seen in fig. 14, 15 and 16 from [5]. Taking this into account, the total cost will be $r(2 \cdot \text{Prepare} + \text{Select} + R)$, where R stands for the reflection in each step.

D.2 How to compute its cost

The circuit implementing the Select operator is depicted in the above-mentioned figure 14 [5]. It will require the use of 3 QROM applications of size $O(N)$, and $2\lceil\log_2 N\rceil$ controlled swaps (Fredking gates) each requiring one T gate. So, the total T-gate cost is $12N + 8\lceil\log_2 N\rceil - 14$.

Subprepare is the main building block for Prepare, and it is depicted in figure 15 in [5]. It uses one QROM, with AND complexity $3M^D - 1 = 3N/2 - 1$, so T complexity $6N - 4$. The 3 is due to the three possible combinations that can appear in $|U\rangle$ and $|V\rangle$, whereas M^D is due to register $|p\rangle$ having $D\lceil\log_2 M\rceil = \lceil\log_2 N/2\rceil$ qubits. Appart from the QROM, Subprepare contains $3 + \lceil\log_2 N/2\rceil = 2 + \lceil\log_2 N\rceil$ controlled swaps (each requiring a Toffoli gate or 4 T gates); two comparison test between 2 μ-sized registers; and finally operators $\text{Uniform}_M^{\otimes D}$ and Uniform_3.

The Uniform operators prepare an uniform superposition over the first L basis states, and is analyzed in figure 12 in [5]. Since in particular we are using Uniform_3 and $\text{Uniform}_M^{\otimes D}$, this will require $8\lceil\log_2 L\rceil + O(\log_2 \epsilon_{SS}^{-1}) = 8\lceil\log_2 3\rceil + O(\log \epsilon_{SS}^{-1})$ T gates in the first case, and $8D\log M + O(\log \epsilon_{SS}^{-1}) = 8\lceil\log_2 N\rceil - 8 + O(\log \epsilon_{SS}^{-1})$ in the second. The $O(\log \epsilon_{SS}^{-1})$ term stands for 2 rotations R_z in each Uniform operator. Overall, Subprepare requires $6N + 12\lceil\log_2 N\rceil + 10\mu + 16\lceil\log_2 \epsilon_{SS}^{-1}\rceil$ T gates.

The Prepare operator can be seen in figure 16 in [5]. It requires another $\text{Uniform}_M^{\otimes D}$, at cost $8\lceil\log_2 N\rceil + 8\lceil\log_2 \epsilon_{SS}^{-1}\rceil$; $D\lceil\log_2 M\rceil = \lceil\log_2 N\rceil - 1$ swaps with 4 times as many T gates; 2 multicontrolled Not gates with $\lceil\log_2 N\rceil$ controls each, which can be implemented using $16\lceil\log_2 N\rceil$ T gates [8]; and one sum over $D\lceil\log_2 M\rceil$ qubits.

With the previous, we have everything we need to calculate the total T gate cost accurately.

E Plane and dual wave basis

E.1 Method explanation

When looking for a basis of functions to perform chemical calculations, one is primarily looking for a basis that [6]

1. Leads to a small number of terms in the Hamiltonian.

2. Allows for simple preparation of initial state.

On the Gaussian basis, initial states are easy to prepare using the Hartree-Fock procedure. However, the Hamiltonian may have up to $O(N^4)$ terms.

One idea to avoid having so many terms in the Hamiltonian is to use the plane waves and dual wave basis. The plane wave basis functions have the form

$$\varphi_{\boldsymbol{\nu}}(\boldsymbol{r}) = \sqrt{\frac{1}{\Omega}} e^{i\boldsymbol{k}_{\boldsymbol{\nu}}\cdot\boldsymbol{r}}, \quad \boldsymbol{k}_{\boldsymbol{\nu}} = \frac{2\pi\boldsymbol{\nu}}{\Omega^{1/3}}, \qquad \boldsymbol{\nu} \in [-N^{-1/3}, N^{1/3}]^3 \in \mathbb{Z}^3. \tag{64}$$

In the plane wave basis, the Hamiltonian will take the form [6]

$$H = \underbrace{+\frac{2\pi}{\Omega} \sum_{\substack{(p,\sigma)\neq(q,\sigma') \\ \nu\neq 0}} \frac{c_{p,\sigma}^\dagger c_{q,\sigma'}^\dagger c_{q+\nu,\sigma'} c_{p-\nu,\sigma}}{k_\nu^2}}_{V}$$
$$\underbrace{+\frac{1}{2}\sum_{p,\sigma} k_p^2 c_{p,\sigma}^\dagger c_{p,\sigma}}_{T} \underbrace{-\frac{4\pi}{\Omega} \sum_{\substack{p\neq q; \\ j,\sigma}} \left(\zeta_j \frac{e^{ik_{q-p}\cdot R_j}}{k_{p-q}^2}\right) c_{p,\sigma}^\dagger c_{q,\sigma}}_{U}, \tag{65}$$

$p, q \in [-N^{-1/3}, N^{1/3}]^3$ indexing the momentum. Notice that in this basis the kinetic operator T is diagonal, a property that we will use abundantly.

Fourier transforming (65), we get the dual plane

wave Hamiltonian,

$$H = \underbrace{\frac{1}{2N}\sum_{p,q,\nu,\sigma} k_\nu^2 \cos[k_\nu \cdot r_{q-p}] a^\dagger_{p,\sigma} a_{q,\sigma}}_{T} \underbrace{-\frac{4\pi}{\Omega}\sum_{p,j,\sigma,\nu\neq 0}\left(\frac{\zeta_j \cos[k_\nu \cdot (R_j - r_p)]}{k_\nu^2}\right) n_{p,\sigma}}_{U} + \underbrace{\frac{2\pi}{\Omega}\sum_{(p,\sigma)\neq(q,\sigma');\nu\neq 0}\frac{\cos[k_\nu \cdot r_{p-q}]}{k_\nu^2} n_{p,\sigma} n_{q,\sigma'}}_{V}, \tag{66}$$

with $n_p = a_p^\dagger a_p$, a_p and $a_p^\dagger$ the Fourier transformed annihilation and creation operators, and $\boldsymbol{r_p} = \boldsymbol{p}(\Omega/N)^{1/3}$. We can see that in this basis the potential terms become diagonal, and since the term V only has $\Theta(N^2)$ terms, the number of terms in the Hamiltonian is $O(N^2)$.

In Jordan Wigner mapping, (66) can be represented as

$$\begin{aligned} H = {} & \frac{\pi}{2\Omega}\sum_{\substack{(p,\sigma)\neq(q,\sigma')\\ \nu\neq 0}} \frac{\cos[k_\nu \cdot r_{p-q}]}{k_\nu^2} Z_{p,\sigma} Z_{q,\sigma'} \\ & \sum_{\substack{p,\sigma\\ \nu\neq 0}}\left(\frac{\pi}{\Omega k_\nu^2} - \frac{k_\nu^2}{4N} + \frac{2\pi}{\Omega}\sum_j \frac{\zeta_j \cos[k_\nu \cdot (R_j - r_p)]}{k_\nu^2}\right) Z_{p,\sigma} \\ & + \frac{1}{4N}\sum_{\substack{p\neq q\\ \nu,\sigma}} k_\nu^2 \cos[k_\nu \cdot r_{q-p}](X_{p,\sigma} Z_{p+1,\sigma} ... Z_{q-1,\sigma} X_{q,\sigma} \\ & + Y_{p,\sigma} Z_{p+1,\sigma} ... Z_{q-1,\sigma} Y_{q,\sigma}) + \sum_{\nu\neq 0}\left(\frac{k_\nu^2}{2} - \frac{\pi N}{\Omega k_\nu^2}\right) I. \end{aligned} \tag{67}$$

Depending on the situation, to simulate the Hamiltonian in the most efficient way possible we will jump back and forth between dual and primal representations depending on the operator of the Hamiltonian

$$H = \underbrace{FFFT^\dagger\left(\frac{1}{2}\sum_{\nu,\sigma} k_\nu^2 a^\dagger_{\nu,\sigma} a_{\nu,\sigma}\right) FFFT}_{T} \underbrace{-\frac{4\pi}{\Omega}\sum_{p,j,\sigma,\nu\neq 0}\left(\frac{\zeta_j \cos[k_\nu \cdot (R_j - r_p)]}{k_\nu^2}\right) n_{p,\sigma}}_{U} + \underbrace{\frac{2\pi}{\Omega}\sum_{(p,\sigma)\neq(q,\sigma');\nu\neq 0}\frac{\cos[k_\nu \cdot r_{p-q}]}{k_\nu^2} n_{p,\sigma} n_{q,\sigma'}}_{V} \tag{68}$$

where all the terms are diagonal. To implement this Hamiltonian, we need to perform a Fermionic Fast Fourier Transform (FFFT) [23], an adaptation of the classical Fast Fourier Transform. We cannot use here the Quantum Fourier Transform because we are using the Jordan-Wigner mapping that encodes the value of the qubits not in the amplitudes but the basis.

E.1.1 Trotterization algorithm

The most basic way to use the plane wave approach is to use (68) to simulate a segment of the Hamiltonian simulation procedure

$$\begin{aligned} e^{-iH\delta_t} \approx {} & e^{-i(U+V)\delta_t/2} \cdot \\ & FFFT^\dagger e^{-i(\delta_t/2)\sum_{\nu,\sigma} k_\nu^2 a^\dagger_{\nu,\sigma} a_{\nu,\sigma}} FFFT \\ & \cdot e^{-i(U+V)\delta_t/2} + O(\delta_t^3), \end{aligned} \tag{69}$$

with U and V given in (68). This formulation allows us to perform Hamiltonian simulation and Quantum Phase Estimation. The FFFT will be explained later on in this appendix.

E.1.2 Taylorization 'database' algorithm

Alternatively, we may use the Taylorization procedures from appendix B. Let us start with the 'database' algorithm. To carry it out we need to define how to perform the Prepare(W) and Select(H) operators.

Select(H) is virtually the same as the same preparation method as we describe in appendix D [5], except that in this case we use the notation of p odd or even for up and down spin values:

$$\text{Select}(H)\,|p,q,b\rangle\,|\psi\rangle = |p,q,b\rangle \otimes \begin{cases} Z_p\,|\psi\rangle & p = q \\ Z_p Z_q\,|\psi\rangle & (b=0)\wedge(p\neq q) \\ X_p \vec{Z} X_q\,|\psi\rangle & (b=1)\wedge(p>q)\wedge(p\oplus q=0) \\ Y_p \vec{Z} Y_q\,|\psi\rangle & (b=1)\wedge(p<q)\wedge(p\oplus q=0) \\ |\psi\rangle & (b=1)\wedge(p\oplus q=1) \end{cases}. \tag{70}$$

where $\oplus$ indicates sum modulus 2; and can therefore be implemented at cost $12N + 8\lceil\log_2 N\rceil + O(1)$ T gates.

Since the Prepare(W) method is not specified in the main reference for this appendix [6], we will also use the method from [5].

E.1.3 Taylorization 'on-the-fly' algorithm

In appendix K of [6] it is explained how to use the 'on-the-fly algorithm' in this context, which is similar to what we explained in appendix B [3].

The amplitudes we want to prepare, $W_{p,q,b}$, can be divided in a sum

$$W_{p,q,b} = \sum_{\nu\neq 0} W_{p,q,b,\nu}, \tag{71}$$

where

$$W_{p,q,b} = \begin{cases} \sum_{\nu\neq 0}\left(\frac{\pi}{2\Omega k_\nu^2}-\frac{k_\nu^2}{8N}+ \right. & p=q \\ \left.\frac{\pi}{\Omega}\sum_j \zeta_j \frac{\cos[k_\nu\cdot(R_j-r_p)]}{k_\nu^2}\right) & \\ \frac{\pi}{4\Omega}\sum_{\nu\neq 0}\frac{\cos[k_\nu\cdot r_{p-q}]}{k_\nu^2} & (b=0)\wedge(p\neq q) \\ \frac{1}{4N}\sum_\nu k_\nu^2\cos[k_\nu\cdot r_{p-q}] & (b=1)\wedge(p\oplus q=0) \\ 1 & (b=1)\wedge(p\oplus q=1). \end{cases} \tag{72}$$

If we have to sum over a large number of atoms J, we may also decompose each of the terms in the j sum independently.

Since it is easy to apply phases but not to change the amplitudes of a given state, [6] proposes further dividing each

$$W_{p,q,b,\nu} \approx \zeta \sum_{m=0}^{M-1} W_{p,q,b,\nu,m}; \quad W_{p,q,b,\nu,m}\in\{\pm 1\};$$
$$\zeta = \Theta\left(\frac{\epsilon}{\Gamma t}\right); \quad M\in\Theta\left(\frac{\max_{p,q,b,\nu}|W_{p,q,b,\nu}|}{\zeta}\right). \tag{73}$$

To perform the logic of the on-the-fly algorithm we first have to perform the calculations for the coefficients, which means we need costly arithmetic operations:

$$\text{Sample}(W)\,|p,q,b,\nu\rangle\,|0\rangle^{\otimes\lceil\log_2 N\rceil} \mapsto |p,q,b,\nu\rangle\,|\tilde{W}_{p,q,b,\nu}\rangle\,, \tag{74}$$

with $W_{p,q,b,\nu}$ a binary approximation to $\tilde{W}_{p,q,b,\nu}$.

The complexity will be $O(N^3+\log_2\epsilon_M^{-1})$, where the ϵ_M appears due to the use of Subprepare techniques from [5].

E.2 How to compute its cost

E.2.1 Trotterization algorithm

In this subsection we aim to explain the cost of performing Trotterization using this approach. To do so, we have to compute the cost of the FFFT operator, as well as the number of single qubit rotations in the exponentials and the number of segments required, r.

Let us start with the computation of the cost of FFFT. From [23] it can be seen that the number of gates required to perform an m-mode Fourier Transform are are $(m/2)\lceil\log_2(m/2)\rceil$ single qubit rotations and $(m/2)\lceil\log_2 m\rceil$ F_2 gates. The matrix representation of F_2 in the Jordan-Wigner representation is

$$F_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2^{-1/2} & 2^{-1/2} & 0 \\ 0 & 2^{-1/2} & -2^{-1/2} & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2^{-1/2} & 2^{-1/2} & 0 \\ 0 & 2^{-1/2} & -2^{-1/2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{75}$$

Therefore, we can see that F_2 is the product of a matrix that we will call W with a Control-Z. The gate W works as a Hadamard in the subspace spanned by $\{|01\rangle, |10\rangle\}$. Any gate with the structure of a unitary gate U in that subspace can be constructed as $C-U$ between two C-Nots in the opposite direction. In this case, U is the Hadamard gate, and the controlled-Hadamard gate can be performed using $R_y(\pi/4)$, a C-Not, and $R_y(-\pi/4)$. Therefore, in total F_2 requires two T gates in the Jordan-Wigner representation.

Overall, the FFFT requires $(N/2)\log_2(N/2) = (N/2)(\log_2 N - 1)$ single qubit z-rotations and $(N/2)\log_2(N)$ F_2 gates, as can be seen from figure 1b from [23].

The next step is computing the cost of the exponential rotations in (69). There are $8N$ terms in U, $8N(8N-1)/2$ terms in V and $8N$ terms in T in (68), so the same number of R_z rotations for operators T and U. Notice that in the simulation of $e^{-iV\tau}$ we will need Clifford gates and a single $C-R_z$ rotation per term [30, 51], as it was the case in appendix A.

Finally we want to compute the number of time segments in the Trotter decomposition r. Using the equations 5 and 6 from [55] we can see that the error in each time step is bounded by

$$2\left([T,[T,U+V]]+[(U+V),[T,(U+V)]]\right)\delta_t^3. \tag{76}$$

This, in turn, can be bounded [6] by

$$2(\max_\psi|\langle\psi|T|\psi\rangle|^2\cdot\max_\psi|\langle\psi|U+V|\psi\rangle|+ \max_\psi|\langle\psi|T|\psi\rangle|\cdot\max_\psi|\langle\psi|U+V|\psi\rangle|^2)\delta_t^3. \tag{77}$$

Since there are $r := t/\delta_t$ terms, the Trotter error is

$$\frac{\epsilon_{HS}}{r} \leq 2(\max_\psi|\langle\psi|T|\psi\rangle|^2\cdot\max_\psi|\langle\psi|U+V|\psi\rangle|+ \max_\psi|\langle\psi|T|\psi\rangle|\cdot\max_\psi|\langle\psi|U+V|\psi\rangle|^2)\left(\frac{t}{r}\right)^3. \tag{78}$$

Asymptotically, this means we will take

$$r = \Theta\left(\frac{\eta^2 N^{5/6}t^{3/2}}{\Omega^{5/6}\sqrt{\epsilon_{HS}}}\sqrt{1+\frac{\eta\Omega^{1/3}}{N^{1/3}}}\right). \tag{79}$$

We can find bounds for the expected values of U, V and T, in appendix F [6]. From equation F1

$$\max_{\psi} |\langle\psi|V|\psi\rangle| \leq \frac{2\pi\eta^2}{\Omega}\sum_{\nu\neq 0}\frac{1}{k_\nu^2} = \frac{\eta^2}{2\pi\Omega^{1/3}}\sum_{(\nu_x,\nu_y,\nu_z)\neq(0,0,0)}\frac{1}{\nu_x^2+\nu_y^2+\nu_z^2}, \tag{80a}$$

from F8

$$\max_{\psi} |\langle\psi|U|\psi\rangle| \leq \frac{4\pi\eta}{\Omega}\left(\sum_j \zeta_j\right)\sum_{\nu\neq 0}\frac{1}{k_\nu^2} = \frac{\eta^2}{\pi\Omega^{1/3}}\sum_{(\nu_x,\nu_y,\nu_z)\neq(0,0,0)}\frac{1}{\nu_x^2+\nu_y^2+\nu_z^2}, \tag{80b}$$

and from F10

$$\max_{\psi} |\langle\psi|T|\psi\rangle| \leq \frac{2\pi^2\eta}{\Omega^{2/3}}\nu_{\max}^2. \tag{80c}$$

To end up bounding U and V we need equation F6 [6]

$$\sum_{(\nu_x,\nu_y,\nu_z)\neq(0,0,0)}\frac{1}{\nu_x^2+\nu_y^2+\nu_z^2} \leq 4\pi\left(\sqrt{3}\frac{N^{1/3}}{2}-1\right) + \int_1^{N^{1/3}}\frac{3dz}{z^2}+\int_1^{N^{1/3}}\int_1^{N^{1/3}}\frac{3dxdy}{x^2+y^2} = 4\pi\left(\sqrt{3}\frac{N^{1/3}}{2}-1\right)+3-\frac{3}{N^{1/3}}+ \int_1^{N^{1/3}}\int_1^{N^{1/3}}\frac{3dxdy}{x^2+y^2}. \tag{81}$$

Using this and the previous equations, it is possible to calculate the actual value of r, given t and ϵ_{HS}.

E.2.2 Taylorization 'database' algorithm

Since the Prepare(W) method is not specified in the main reference for this appendix [6], we will also use the method from [5]. As explained in appendix D, the cost for Prepare(W) $6N+40\lceil\log_2 N\rceil+16\lceil\log_2\epsilon_{SS}^{-1}\rceil+10\mu$. Notice that the cost is linear because although there are $O(N^2)$ coefficients, only $O(N)$ are independent. In any case this will be multiplied by $\lambda = O(N^2)$.

Similarly taken from [5] and explained in appendix D the cost of Select(H) can be taken to be $12N + 8\lceil\log_2 N\rceil + O(1)$ T gates, since the implementation proposed in both references ([5] and [6]) is virtually the same.

E.2.3 Taylorization 'on-the-fly' algorithm

Finally, the main cost of the 'on-the-fly' algorithm comes from the Sample(W) operations that compute (72). This will require arithmetic operations as those indicated in table 2.

The main difference here will be calculating the value of λ', that influences the number of segments r. From equation K2 in [5] the Hamiltonian will have the form

$$H = \zeta\sum_{p,q,b,\nu,m} W_{p,q,b,\nu,m}H_{p,q,b} \tag{82}$$

Similarly as in previous appendices, we take

$$\zeta = \frac{\epsilon_H}{\Gamma r}. \tag{83}$$

In contrast to appendix B there is no integral over any volume, so we do not include $\mathcal{V}$ in the denominator; and in contrast to appendix C we do not sum over ρ so there is no division by μ. The main consequence of this form of preparing the initial state is changing the value of λ, that will now be, from eq. K5 in [6]

$$\lambda' = \zeta\sum_{p,q,b,\nu,m}|W_{p,q,b,\nu,m}|, \quad W_{p,q,b,\nu,m}\in\{-1,+1\}. \tag{84}$$

As a consequence, given that $m \in 0,...,M-1$, b can take values 0 and 1 and there are $8N$ values for p, q and ν

$$\lambda' = 2M\zeta(8N)^3. \tag{85}$$

Since

$$M = \frac{\max_{p,q,b,\nu}|W_{p,q,b,\nu}|}{\zeta} \tag{86}$$

we have that

$$\lambda' = 2(8N)^3\max_{p,q,b,\nu}|W_{p,q,b,\nu}| \tag{87}$$

As the sum of the nuclear charges is equal to the number of electrons $\sum_j\zeta_j = \eta$, we can bound $\max_{p,q,b,\nu}|W_{p,q,b,\nu}|$ as the maximum of 1 (the identity term);

$$\frac{\pi}{2\Omega k_\nu^2}-\frac{k_\nu^2}{8N}+\frac{\pi}{\Omega}\sum_j\zeta_j\frac{\cos[k_\nu\cdot(R_j-r_p)]}{k_\nu^2} \leq \frac{\pi}{2\Omega k_\nu^2}-\frac{k_\nu^2}{8N}+\frac{\pi\eta}{\Omega k_\nu^2} \leq \frac{\pi}{2\Omega k_\nu^2}+\frac{\pi\eta}{\Omega k_\nu^2}-\frac{k_\nu^2}{8N} = \frac{(2\eta+1)\pi}{2\Omega k_\nu^2}-\frac{k_\nu^2}{8N}; \tag{88a}$$

$$\frac{\pi}{4\Omega}\frac{\cos[k_\nu\cdot r_{p-q}]}{k_\nu^2} \leq \frac{\pi}{4\Omega k_\nu^2}; \tag{88b}$$

or

$$\frac{k_\nu^2}{4N}\cos[k_\nu\cdot r_{p-q}] \leq \frac{k_\nu^2}{4N}. \tag{88c}$$

Since the smallest value of $|k_\nu|$ for $\nu \neq 0$ is $k_\nu = 2\pi/\Omega^{1/3}$, and the largest is $k_\nu^2 = 3 \times \frac{(2\pi)^2 N^{2/3}}{\Omega^{2/3}}$

$$\max_{p,q,b,\nu} |W_{p,q,b,\nu}| \leq \max\left[\frac{(2\eta+1)}{8\pi\Omega^{1/3}} - \frac{\pi^2}{2N\Omega^{2/3}}, \frac{1}{8\pi\Omega^{1/3}}, \frac{6\pi^2}{N^{1/3}\Omega^{2/3}}\right], \tag{89}$$

Provided that the first option is the largest,

$$\lambda' \leq (8N)^3 \left(\frac{(2\eta+1)}{4\Omega^{1/3}\pi} - \frac{\pi^2}{N\Omega^{2/3}}\right). \tag{90}$$

Now we want to compute the number of arithmetic operations in the Prepare(w) operation. $p = q$ case of (72):

1. Calculating k_ν and r_p requires three multiplications each, one for each coordinate component, with $n = \lceil \log_2 N^{1/3} \rceil$.
2. There are three subtraction for each value of j in $R_j - r_p$ and another $r_{p-q} = r_p - r_q$, with $n = \lceil \log_2 N^{1/3} \rceil$.
3. Computing k_ν^2 requires 3 multiplications and 2 additions.
4. Calculating the product within the cosines costs three multiplications of length $n = \lceil \log_2 N^{1/3} \rceil$, and two sums between those terms.
5. One of the fastest ways to compute the cosine is to use the CORDIC algorithm [68], which requires a prefactor division (if expanded to a fixed order) and 2 sums per order since divisions by powers of two can be performed virtually.
6. We have to sum J cosine computations.
7. We have to divide or multiply such sum of cosines by a constant, and k_ν^2. Costs up to $\approx 3{\cdot}21 \log^2 N$.

Thus, the T-gate cost of this first calculation is $\approx J\left[\frac{35o}{2} + 63 + \frac{2o}{\log_2 N}\right] \log_2^2 N$, where J is the number of values of j, that indexes the atoms.

For $(b = 0) \wedge (p \neq q)$ and $(b = 1) \wedge (p \oplus q = 0)$:

1. We can reuse the previously calculated values of k_ν, k_ν^2 and compute r_q (3 multiplications) and r_{p-q} (3 subtractions).
2. We can perform the dot product in the cosine with 3 multiplications and 2 sums
3. Similarly, we have to perform a cosine calculation via the CORDIC algorithm again.
4. Finally we perform a multiplications and a division (by k_ν^2)

To perform the case $b = 1 \wedge (p+q = 0 \mod 2)$ we can reuse the cosine result from the previous point, as well as the k_ν^2 value, so we only need two multiplications.

E.3 How to adapt the Hamiltonian simulation to control the direction of the time evolution

E.3.1 Trotterization method

In the Phase Estimation protocol we should be controlling such rotations depending on the control ancilla qubits. However, since they are R_z rotations and $XR_z(\alpha)X = R_z(-\alpha)$ we can actually use a formulation similar to [5] where the mapping is $|1\rangle|\phi\rangle \to e^{i\phi}|1\rangle|\phi\rangle$ and $|0\rangle|\phi\rangle \to e^{-i\phi}|0\rangle|\phi\rangle$ (except for the first segment, but this is a minor cost). To control between both rotations we use C-Nots which change the direction of the Z rotation [74].

E.3.2 Taylorization methods

Adapting the Hamiltonian simulation for its use in Quantum Phase Estimation can be done as in appendix B.3. The cost can be therefore calculated in the same way.

F Trotter simulation: tighter bounds

In the previous appendix E we have explained how to perform Trotter simulation in plane waves. However, the bounds provided by (77) are somewhat loose, so the number of steps needed to achieve the same error are lower than required. Similarly happens for the methods covered in appendix A. In this appendix we give tighter bounds for the second order Hamiltonian simulation deterministic Trotter operator. We aim to approximate $e^{iH\delta_t}$, for $H = \sum_{\gamma=1}^{\Gamma} w_\gamma H_\gamma$, with

$$\mathcal{S}_2(H;\delta_t) = \left(\prod_{\gamma=1}^{\Gamma} e^{\frac{it}{2} w_\gamma H_\gamma}\right)\left(\prod_{\gamma=\Gamma}^{1} e^{\frac{it}{2} w_\gamma H_\gamma}\right). \tag{91}$$

This general expression will reduce, for the plane wave basis, to (69)

$$e^{-iH\delta_t} \approx e^{-i(U+V)\delta_t/2} \cdot FFFT^\dagger e^{-i(\delta_t/2)\sum_{\nu,\sigma} k_\nu^2 a_{\nu,\sigma}^\dagger a_{\nu,\sigma}} FFFT \cdot e^{-i(U+V)\delta_t/2} + O(\delta_t^3). \tag{92}$$

The error in this expression will be

$$\|e^{iH\delta_t} - \mathcal{S}_2(H;\delta_t)\| \leq W_2 \delta_t^3, \tag{93}$$

for $\delta_t = t/r$ and W_2 a commutator expression. Since in plane waves, operators U and V commute, in a Hamiltonian $H = T + U + V$ we only have to care about commutators $[[T, U+V], T]$ and $[[T, U+V], U+V]$. This can be better seen in the dual basis, where

$$V = \sum_{p \neq q} V_{pq} n_p n_q \tag{94}$$

and

$$U = \sum_p U_p n_p = \sum_p U_p n_p n_p, \tag{95}$$

for n_p the occupancy fermionic operator. Since the n_p operators commute with each other, so do U and V. Consequently, Ref. [63] proposes to write $H = T + \bar{V}$ with

$$\bar{V} := U + V = \sum_{p,q} \bar{V}_{p,q} n_p n_q. \tag{96}$$

One additional insight to bound the commutator W_2 as tightly as possible is to restrict our space to the space of η electrons. Usually, the error has been described in terms of the spectral norm distance, that is, in other words $\|H\|_2 = \max_\psi \| \langle\psi|H|\psi\rangle \|$. However, this takes into account states ψ that do not live in the subspace of η electrons, potentially leading to a higher norm and a looser bound. To remedy this, one can instead use the 'fermionic seminorm', defined as

$$\|H\|_\eta = \max_{\phi,\psi\in\mathcal{H}_\eta} \| \langle\phi|H|\psi\rangle \|, \tag{97}$$

for $\mathcal{H}_\eta$ the Hilbert subspace with η electrons. While this seminorm fulfills many properties of norms such as the triangle inequality, it is not a norm because some operators can evaluate to 0 without being operator 0 in the full Hilbert space, for example $\|n_p n_q\|_{\eta=1} = 0$.

Using the fermionic seminorm, we express the commutator error bound W_2 as [48]

$$W_2 \leq \frac{1}{12}\|[[T, U+V], T]\|_\eta + \|[[T, U+V], U+V]\|_\eta. \tag{98}$$

Furthermore, it is possible to bound each of the two terms independently as ([63] and appendix A in [48]):

$$\|[[T,\bar{V}],T]\|_\eta \leq 4\|T\|_2^2\|\bar{V}\|_{\max}\eta(4\eta+1) \tag{99}$$

$$\|[[T,\bar{V}],\bar{V}]\|_\eta \leq 12\|T\|_2\|\bar{V}\|_{\max}^2\eta^2(2\eta+1), \tag{100}$$

collectively known as the SHC bound, which scales as $O(N^3)$ with the number of basis functions N. Other bounds exist too (see sections 3 and 4, and table 1 in [48]), and shall be included in future updates to the library.

From equation 8 in [6], we also know that

$$\|U+V\|_{\max} \leq \frac{4\pi}{\Omega}\frac{\Omega^{2/3}}{4\pi^2}\sum_i \zeta_i = \frac{\Omega^{1/3}\eta}{\pi}, \tag{101}$$

while $\|T\|_2$ can be bounded as we did in (80c). From this, and the implementation cost of (69) that we discussed in appendix E.2, we can obtain an even lower cost of the Trotter simulation.

G Sparsity and low rank factorization

G.1 Method explanation

In the previous appendix we have seen that using carefully crafted Prepare and Select operators, it is possible to lower the complexity of the Quantum Phase Estimation. However, this came at the cost of having to use plane waves or similar basis sets. The method proposed in this appendix allows to leverage QROM techniques while working in arbitrary basis [5, 11]. The other main consideration of this article is how to leverage the sparsity and a low rank factorization of the Hamiltonian to lower the complexity of the algorithm.

Let us start by the second aspect, the low rank tensor factorization. We know that we can write the Hamiltonian in the second quantization in the following form

$$\begin{aligned} H &= \sum_{\sigma\in\{\uparrow,\downarrow\}}\sum_{p,q=1}^{N/2} h_{pq}a_p^\dagger a_q \\ &\quad + \frac{1}{2}\sum_{\alpha,\beta\in\{\uparrow,\downarrow\}}\sum_{p,q,r,s=1}^{N/2} h_{pqrs}a_{p,\alpha}^\dagger a_{q,\beta}^\dagger a_{r,\beta}a_{s,\alpha} \\ &= \sum_{\sigma\in\{\uparrow,\downarrow\}}\sum_{p,q=1}^{N/2} T_{pq}a_p^\dagger a_q \\ &\quad + \sum_{\alpha,\beta\in\{\uparrow,\downarrow\}}\sum_{p,q,r,s=1}^{N/2} V_{pqrs}a_{p,\alpha}^\dagger a_{q,\alpha}a_{r,\beta}^\dagger a_{s,\beta} \end{aligned} \tag{102}$$

The coefficients h_{pq} and h_{pqrs} are efficiently computable integrals. On the previous equation, the ordering $a^\dagger a^\dagger aa$ is called the 'physics notation' whereas the second ordering, $a^\dagger aa^\dagger a$ follows the chemists convention and will be the one we will use because it allows us to perform the factorization. Notice that T_{pq} and V_{pqrs} are real and have symmetries $p \leftrightarrow q$, $r \leftrightarrow s$ and $pq \leftrightarrow rs$. Notice also that the one-body operator changes as a result of the swapping of a_p and $a_p^\dagger$ and their anticommutation in the two-body term, and so does the sign of the latter.

Since V is a 4-rank tensor, with indices ranging from 0 to $N/2-1$, we can transform it to a $N^2/4 \times N^2/4$ matrix called W, with composite indices pq and rs, and symmetric and positive definite. Diagonalizing W we get,

$$Wg^{(l)} = w_l g^{(l)}; \quad W = \sum_{l=1}^{L} w_l g^{(l)}(g^{(l)})^T, \tag{103}$$

where $g^{(l)}$ denotes the l-th eigenvector, with eigenvalue w_l, and entries $g_{pq}^{(l)}$.

Let us denote the rank with L. If W where full rank, $L = N^2/4$. However, in most cases and due to Coulomb interaction being a two-body interaction, the rank will be $L = O(N)$. Now, we can rewrite

$$\begin{aligned} &\sum_{\alpha,\beta\in\{\uparrow,\downarrow\}}\sum_{p,q,r,s=1}^{N/2} V_{pqrs}a_{p,\alpha}^\dagger a_{q,\alpha}a_{r,\beta}^\dagger a_{s,\beta} \\ &= \sum_{l=1}^{L} w_l \left(\sum_{\sigma\in\{\uparrow,\downarrow\}}\sum_{p,q=1}^{N/2} g_{pq}^{(l)}a_{p,\sigma}^\dagger a_{q,\sigma}\right)^2. \end{aligned} \tag{104}$$

From the right-hand side of the equation we can see that there are $O(LN^2) = O(N^3)$ independent coefficients. In fact, due to the symmetry $p \leftrightarrow q$ there are $1/2 \cdot N/2(N/2-1)$ terms off diagonal, and when $p = q$ there are $N/2$ additional free coefficients. Therefore, in total there are $N^2/8 + N/4$ independent terms for each value of l. Further factorization is possible [39, 50, 69], but this work is not covered in this appendix.

As in the previous article, we do not attempt to perform phase estimation over $e^{\pm iH}$ but rather over $e^{\pm i \arccos(E_k/\lambda)}$, which is the phase produced by one step of the qubitization quantum walk. Also as in the previous article, this method uses Jordan-Wigner mapping too.

We have to explain how to perform operators Prepare and Select. Let us start with the former. The state we want to prepare is the following

$$|\psi\rangle = |0\rangle|+\rangle|0\rangle \sum_{p,q,\sigma} \sqrt{\frac{|T_{pq}|}{\lambda}} |\theta_{pq}^{(0)}\rangle |0\rangle |p,q,\sigma\rangle |0\rangle + \\ + \sum_l \sqrt{\frac{w_l}{\lambda}} |l\rangle |+\rangle |+\rangle \otimes \\ \otimes \sum_{p,q,r,s,\alpha,\beta} \sqrt{|g_{pq}^{(l)} g_{rs}^{(l)}|} |\theta_{pq}^{(l)}\rangle |\theta_{rs}^{(l)}\rangle |p,q,\alpha\rangle |r,s,\beta\rangle . \tag{105}$$

Here, $\theta_{pq}^{(l)}$ indicates the sign of each term, and are defined as

$$\theta_{pq}^{(l)} = \begin{cases} 0, & T_{pq} > 0, \\ 1, & T_{pq} < 0, \end{cases} \qquad \theta_{pq}^{(l)} = \begin{cases} 0, & g_{pq}^{(l)} > 0. \\ 1, & g_{pq}^{(l)} < 0. \end{cases} \tag{106}$$

We can see that the first register selects between the T terms (for state $|0\rangle$) and each of the L terms for $g^{(l)}$. The second and third register use $|+\rangle$ to select between $\mathbf{1}$, and $Z_{p,\sigma}$, $Z_{p,\alpha}$ and $Z_{q,\beta}$, whenever $p = q$ or $r = s$ respectively. Additionally, depending on whether $p > q$ or $p < q$ we apply $X_{p,\sigma}\vec{Z}X_{q,\sigma}$ or $Y_{p,\sigma}\vec{Z}Y_{q,\sigma}$ respectively.

The number of coefficients to fix is $(L+1)(N^2/8 + N/4)$, so the complexity will be $O(N^3 + \log_2 \epsilon_\mu^{-1})$, where the μ appears due to the use of Subprepare techniques from [5]. To perform the preparation, we follow this steps

1. Starting from the state $|0\rangle$, prepare a superposition over the first register

$$\left(|0\rangle \sqrt{\sum_{p,q} \frac{2|T_{pq}|}{\lambda}} + 2 \sum_l \sqrt{\frac{w_l}{\lambda}} |l\rangle \sum_{p,q} |g_{p,q}^{(l)}| \right) \otimes \\ \otimes |0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle . \tag{107}$$

   If we allow for error ϵ_{SS}, the complexity of this step, in terms of T-gates using the QROM is $4L + 4\mu + 14\lceil \log_2 L \rceil + 8\lceil \log_2 \epsilon_{SS}^{-1} \rceil$ [5]. The ϵ_{SS}^{-1} dependence is due to the Uniform operator preparation, that requires to use two controlled Z rotations, at cost $4\lceil \log_2 \epsilon_{SS}^{-1} \rceil$ each. On the other hand, the Uniform preparation requires $10\lceil \log_2 L \rceil$ T gates as can be seen from figure 12 in [5], which has to be added to $4\lceil \log_2 L \rceil$ T-gates due to the controlled-swap operations in Subprepare. The value of μ can be taken from equation 36 in [5].

2. Perform a Hadamard in the second register and another on the third, controlled on the first register being $|l > 0\rangle$.

$$\left(|0\rangle|+\rangle|0\rangle \sqrt{\sum_{p,q} \frac{2|T_{pq}|}{\lambda}} \right. \\ \left. + 2 \sum_l \sqrt{\frac{w_l}{\lambda}} |l\rangle|+\rangle|+\rangle \sum_{p,q} |g_{p,q}^{(l)}| \right) \otimes \\ \otimes |0\rangle|0\rangle|0\rangle|0\rangle . \tag{108}$$

   The cost of this step is negligible compared with the following one, and can be performed using a multicontrolled Hadamard.

3. Prepare a superposition over register six with amplitudes $\sqrt{|T_{pq}|}$ if $|l = 0\rangle$ or $\sqrt{|g_{pq}^{(l)}|}$ if $|l > 0\rangle$.

$$|0\rangle|+\rangle|0\rangle \sum_{p,q,\sigma} \sqrt{\frac{|T_{pq}|}{\lambda}} |0\rangle|0\rangle|p,q,\sigma\rangle|0\rangle + \\ + \sqrt{2} \sum_l \sqrt{\frac{w_l}{\lambda}} |l\rangle|+\rangle|+\rangle \otimes \\ \otimes \sum_{p,q,\alpha} \sqrt{|g_{p,q}^{(l)}|} \sqrt{\sum_{r,s} |g_{r,s}^{(l)}|} |0\rangle|0\rangle|p,q,\alpha\rangle|0\rangle . \tag{109}$$

   This step and the following have the largest complexities, since we need to use the unary iterator and Subprepare circuit of [5]. We have to iterate over L, p, and q, and that gives a Toffoli complexity of $(L+1)N^2/4 - 1$ plus the cost of the comparison and the controlled swaps from Subprepare.

4. For $|l > 0\rangle$, prepare weights $\sqrt{|g_{rs}^{(l)}|}$ in register 7.

$$|0\rangle|+\rangle|0\rangle \sum_{p,q,\sigma} \sqrt{\frac{|T_{pq}|}{\lambda}} |0\rangle|0\rangle|p,q,\sigma\rangle|0\rangle + \\ + \sum_l \sqrt{\frac{w_l}{\lambda}} |l\rangle|+\rangle|+\rangle \otimes \\ \otimes \sum_{p,q,r,s,\alpha,\beta} \sqrt{|g_{p,q}^{(l)} g_{r,s}^{(l)}|} |0\rangle|0\rangle|p,q,\alpha\rangle|r,s,\beta\rangle . \tag{110}$$

In this step the Toffoli complexity is also $LN^2/4$ plus the cost of the compare and controlled swaps.

5. Finally, use the QROM to output $|\theta^{(l)}_{pq}\rangle$ and $|\theta^{(l)}_{rs}\rangle$ in registers four and five.

$$|0\rangle|+\rangle|0\rangle\sum_{p,q,\sigma}\sqrt{\frac{|T_{pq}|}{\lambda}}|\theta^{(0)}_{pq}\rangle|0\rangle|p,q,\sigma\rangle|0\rangle + \\ +\sum_{l}\sqrt{\frac{w_l}{\lambda}}|l\rangle|+\rangle|+\rangle\otimes \\ \otimes\sum_{p,q,r,s,\alpha,\beta}\sqrt{|g^{(l)}_{pq}g^{(l)}_{rs}|}|\theta^{(l)}_{pq}\rangle|\theta^{(l)}_{rs}\rangle|p,q,\alpha\rangle|r,s,\beta\rangle\,. \tag{111}$$

To alleviate the cost of this procedure we follow three procedures:

1. Leverage the $p\leftrightarrow q$ symmetry in T_{pq} and $g^{(l)}_{pq}$, which divides the cost by half. This can be done preparing initially

$$\sqrt{2}\sum_{p>q}\sqrt{|g^{(l)}_{pq}|}|p,q,\alpha\rangle+\sum_{p}\sqrt{|g^{(l)}_{pp}|}|p,p,\alpha\rangle\,. \tag{112}$$

Then, one can use the second register, in state $|+\rangle$ to swap $|p\rangle$ and $|q\rangle$ when $p\neq q$ or to apply either $\mathbf{1}$ or $Z_{p,\sigma}$ when $p=q$. This means that in step 3 we will have to prepare $(L+1)(N^2/8+N/4)$ entries, and in step 4, $L(N^2/8+N/4)$.

2. We can also reduce the preparation cost in the QROM by performing the comparison between the probability $|\text{keep}_j\rangle$ and an ancilla in uniform superposition, at the same time for all $l\in(0,...,L)$. The controlled swap between the register $|j\rangle$ and $|\text{alt}_j\rangle$ can also be performed for all values of l simultaneously.

3. The dominant cost is outputting $(2L+1)(N^2/8+N/4)$ qubits using the QROM [5]. The outputs will have a size $M=\lceil\log_2 N^2\rceil+\lceil\log_2\epsilon^{-1}_{QPE}\rceil+O(1)$ where $\lceil\log_2 N^2\rceil$ is the size of $|\text{alt}\rangle$ and $\mu=\lceil\log_2\epsilon^{-1}_{QPE}\rceil+O(1)$ $|\text{keep}\rangle$, the size of the probability register. The key aspect of this third point is substituting the QROM of [5] by another from [46] which allows to trade some gate complexity by space complexity. We will call it QROAM. Calling also $d=(2L+1)(N^2/8+N/4)$ the number of entries we must look in the QROAM (including steps 3, 4 and 5), and $k=2^n$ an arbitrarily chosen power of 2. Then the complexity of computing the QROAM is $\lceil d/k_c\rceil+M(k_c-1)$ uncomputing it in Prepare† is $\lceil d/k_u\rceil+k_u$, where the k_c and k_u in compute and uncompute respectively can be different.

   As an aside, we can indicate that if we were to use dirty ancillae (anciallae that is already being used for other purposes) the cost would be $2\lceil d/k\rceil+4M(k-1)$ and $2\lceil d/k\rceil+4k$ for compute and uncompute respectively.

   Since the largest bottleneck is in the number of Toffolis required, we will focus on minimizing that variable. This means taking $k\approx\sqrt{d/M}$ for compute and $k\approx\sqrt{d}$ for the uncompute step, what means a cost of $2\sqrt{dM}$ and $2\sqrt{d}$ respectively, giving a total cost of $2\sqrt{d}(\sqrt{M}+1)$. Since we have chosen $d\approx LN^2/8$ and $M\approx\lceil\log_2(N^2)\rceil+\mu$, this means an overall cost $\sqrt{LN^2(\lceil\log_2(N^2)\rceil+\mu)/2}$ and half as many ancillae. Since $L=O(N)$, the number of Toffolis is $O(N^{3/2}\sqrt{\lceil\log_2 N\rceil+\mu})$.

A technical detail is that since the QROAM requires a continous output register, we will compute a single continous register for (l,p,q)

$$s'=l(N^2/8+N/4)+p(p+1)/2+q \tag{113}$$

The second operator we have to explain is Select, which is decomposed in two, Select$_1$ and Select$_2$ [11], performed again similarly as is done in appendix D [5]. The cost of this procedure is not dominant, as it will have complexity $O(N)$. From the representation of Select$_1$ in Figure 1 of [11], we can see that we need two QROM applications, as well as 2 equality comparisons.

Apart from the implementation of Prepare and Select, some other minor costs to have in mind are

- The cost in Select of each ranged operation is N, and each inequality test is $\lceil\log_2 N\rceil$. Since these operations have to be performed twice for (p,q) and again twice for (r,s), the total cost is $4N+4\lceil\log_2 N\rceil$.

- In the Prepare operator we have to initially prepare superpositions over $l\leq L$, $q\leq p<N/2$, $s\leq r<N/2$. We propose doing this by using the Uniform routine from the previous appendix (figure 12 in [5]). The initial uniform superposition over l requires $8\log_2 L+8\log_2\epsilon^{-1}_{SS}$ T gates. Enforcing an uniform superposition (in the Subprepare method) where $p\geq q$ requires a different method. We will slightly modify the suggestion of [11] to control the number of Amplitude Amplification steps. We do this by implementing the Uniform protocol both for p and q independently, and then adding an ancilla to check whether $p\geq q$. The success probability will be $\frac{N^2/8+N/4}{N^2/4}$ which approaches 1/2 from above. Since we cannot straightforwardly amplify that we add a second ancilla with success amplitude $\frac{1}{2}\sqrt{\frac{N^2/4}{N^2/8+N/4}}$. As a consequence, the product of success probabilities will be 1/2 that corresponds to a Grover's $\theta=\pi/6$ which can be amplified to amplitude 1 with a single step. So this step requires 2 Uniform$_{N/2}$ procedures and 1 ancilla rotation, to be performed thrice: preparation and

twice for Grover step. This second procedure has to be repeated twice to account for r and s too.

- The inequality test in state preparation has cost μ Toffolis due to the μ bits in $|\text{keep}\rangle$; and the same number of gates as qubits needed in the swap gate. We have to perform swap gates in the preparation procedure in the QROM where the register $|l,p,q\rangle$ has size $\lceil \log_2 L\rceil + 2\lceil \log_2(N/2)\rceil$. Then, we have to perform the same swap for $|r,s\rangle$ controlling on $l>0$, with registers of size $2\lceil \log_2(N/2)\rceil$. This means a Toffoli cost $\mu + \lceil \log_2 L\rceil + 4\lceil \log_2 N/2\rceil$. Here $\mu \approx \left\lceil \log\left(\frac{2\sqrt{2}\lambda}{\epsilon_{QPE}}\right)\right\rceil$.
- For state preparation remember that we only prepare those states that have $p>q$ and then use a controlled swap. These controlled swap for (p,q) and (r,s) cost $2\lceil \log_2 N/2\rceil$ Toffolis.
- The arithmetic operations for computing (113) require $2(\lceil \log_2 N/2\rceil)^2$ Toffolis gates.

In any case, the leading cost of the model is $\sqrt{LN^2(\log(N^2)+\mu)/2}$ Toffoli gates due to the QROAM. We can further reduce the cost by increasing the sparsity of the V operator, by zeroing all the terms $|V_{p,q,r,s}|<c$. Choosing c should be done in a way that does not affect the final error ΔE, as will be done choosing L too. To do that, the main aspect is substituting the QROM indexing

$$\frac{1}{\sqrt{d}}\sum_{j=1}^{d}|j\rangle\,|\text{alt}_j\rangle\,|\text{keep}_j\rangle \tag{114}$$

by another

$$\frac{1}{\sqrt{d}}\sum_{j=1}^{d}|j\rangle\,|\text{ind}_j\rangle\,|\text{alt}_j\rangle\,|\text{keep}_j\rangle \tag{115}$$

where ind_j indicates the j-th non-zero index, and d the number of non-zero terms in each case. This means that the swapping must now be performed between ind_j and alt_j. In this case we cannot simplify $\lceil d/k_c\rceil + (k_c-1)M + \lceil d/k_u\rceil + k_u$ with $M = \mu + 2\log_2 N + 2 \approx \log_2(N^2)+\mu$, directly to $\sqrt{LN^2(\log(N^2)+\mu)/2}$. The 2 in M is because we have to choose between T_{pq}, $g^{(l)}_{pq}$ and $g^{(l)}_{rs}$.

G.2 How to compute its cost

Notice that in contrast to other appendices, this calculation was already present in the original article [11] so the method to compute the cost is not our contribution. Only the automatisation of the computation is.

1. Steps 1 and 2 in state preparation can be performed using a QROM for L values and a multicontrolled-Hadamard gate respectively.
2. The largest cost in each step is the use of the QROAM for steps 3, 4 and 5, that as we saw is $\lceil d/k_c\rceil + M(k_c-1) - \lceil d/k_c\rceil + k_c$ Toffolis. It takes into account both the Prepare and Prepare† operators. We also use this step to prepare step 5, registers $|\theta_{pq}\rangle$ and $|\theta_{rs}\rangle$.
3. Here
   - $d = (2L+1)(N^2/8+N/4)$, as we take into account both steps 3 and 4 at the same time,
   - L is the rank of W. If W were full rank, $L = N^2/4$, but since W has a lot of structure $L = O(N)$.
   - $k_c \approx \sqrt{d/M}$ (closest power of 2),
   - $k_u \approx \sqrt{d}$ (closest power of 2),
   - $M = \log_2 N^2 + \mu$,
   - and $\mu \approx \left\lceil \log\left(\frac{2\sqrt{2}\lambda}{\epsilon_{QPE}}\right)\right\rceil$,
4. Each step requires to use Select once, at cost $4N + 4\lceil \log_2 N\rceil$.
5. At each Prepare we have to use Uniform three times: for l (accounted for in point 1 of this list), and two copies of $\tilde{s}$ for (p,q) and (r,s).
6. Other minor contributions of the Subprepare circuit (see [5]) include a μ-bit comparison and a $\log_2(LN^2/4)$-bit controlled swap.
7. The calculation of (113), which is carried out for pairs (p,q) and (r,s) can be done from the value of $\tilde{s}$ with 2 multiplications and three multiplications.
8. We need to perform amplitude amplification to prepare Uniform superposition over $p\geq q$ and $r\geq s$. This requires 6 Uniform$_{N/2}$ (for p and q and the three times of Amplitude Amplification), thrice an arbitrary rotation of the ancilla, thrice comparison between registers $|p\rangle$ and $|q\rangle$, and 2 Multi-controlled Z; and similarly for r and s respectively.

H Interaction picture

H.1 Method explanation

Although in previous appendices we have explored both the plane wave and Gaussian basis, there are two characteristics we have maintained constant over all the previous methods: all simulations were done in the Schrödinger picture and second quantization. These changes in later articles [7, 44, 62], and in this appendix, we present how the interaction picture can help make more efficient Hamiltonian Simulation algorithms [44].

Let us recall that the Schrödinger picture time evolution is dictated by the solution to the Schrödinger equation

$$\partial_t |\psi(t)\rangle = -iH(t)|\psi(t)\rangle \tag{116}$$

what implies that

$$|\psi(t)\rangle_S = e^{-iHt/\hbar}|\psi(0)\rangle, \tag{117}$$

whenever the Hamiltonian is time independent. We can see that in this case it is the state the one that evolves in time.

On the other hand we have the Heisenberg picture, where the dynamics are included in the operators. As such we have

$$\frac{d}{dt}A(t) = \frac{i}{\hbar}[H, A(t)] + \left(\frac{\partial A}{\partial t}\right)_H. \tag{118}$$

If the Hamiltonian is time independent this becomes

$$A(t)_H = e^{iHt/\hbar}A(0)e^{-iHt/\hbar}. \tag{119}$$

An intermediate option is to choose the interaction or Dirac picture, where both the state and the operators become time dependent. In this case we divide the Hamiltonian in two parts $H_S = H_{S,0}+H_{S,1}$, where $H_{S,1}$ carries the complexity and time dependence of the Hamiltonian. Then, the quantum state will evolve as

$$|\psi\rangle_I = e^{iH_{S,0}t/\hbar}|\psi(0)\rangle \tag{120}$$

and the operators will evolve as

$$A(t)_I = e^{iH_{S,0}t/\hbar}A(0)e^{-iH_{S,0}t/\hbar}. \tag{121}$$

In particular

$$H(t)_I = e^{iH_{S,0}t/\hbar}H_{S,1}e^{-iH_{S,0}t/\hbar}. \tag{122}$$

If the Hamiltonian is time-independent, we can evolve the state using e^{-iHt}, but if it is time-dependent, there is no closed expression in general. The time evolution operator is

$$U(t) = \lim_{r\to\infty}\prod_{j=1}^{r} e^{-iH(t(j-1)/r)\tau} := \mathcal{T}e^{-i\int_0^t H(s)ds}. \tag{123}$$

The authors of [44] explore two topics. In the first place, they build a time-dependent Hamiltonian simulation algorithm that is based on synthesizing a Dyson series. The second part of the article analyses how to apply the previous algorithm to simulate a Hamiltonian in the interaction picture. In particular, this allows us to simulate $e^{-i(H_{S,0}+H_{S,1})t}$ using

$$O(\lambda_1 t \text{poly}\log((\lambda_0+\lambda_1)t/\epsilon)) \tag{124}$$

queries to an oracle

$$(\langle 0|_a \otimes \mathbf{1}_s)O_1(|0\rangle_a \otimes \mathbf{1}_s) = \frac{H_{S,1}}{\lambda_1}, \tag{125}$$

and a similar amount of $e^{-iH_{S,0}\tau}$ queries with $\tau = O(\lambda_1^{-1})$. Here we were taking $\lambda_0 \geq ||H_{S,0}||$ and $\lambda_1 \geq ||H_{S,1}||$. If we had used the Schrödinger picture we would have instead needed

$$O((\lambda_0+\lambda_1)t\text{poly}\log((\lambda_0+\lambda_1)t/\epsilon_{HS})) \tag{126}$$

queries to oracles O_0 and O_1 of the form of (125). If $||H_{S,0}|| \gg ||H_{S,1}||$, and the complexity of applying $e^{-iH_{S,0}t}$ is similar to O_1, the interaction picture algorithm is advantageous.

Finally, the article applies the algorithm to the generalized Hubbard model and the electronic Hamiltonian, with a final complexity $\tilde{O}(N^2t)$.

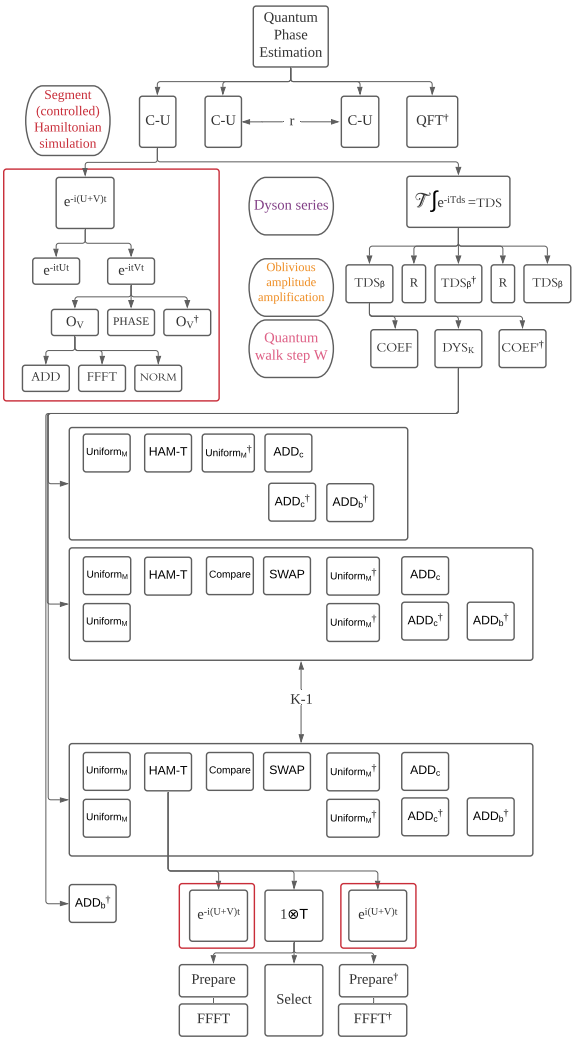

Figure 8: Abstraction level decomposition of the interaction picture protocol of [44]. The boxes in red represent the same protocol, only decomposed for one of them.

In contrast with previous algorithms, we cannot approximate $U(t)$ with a Taylor series unless $[H(t), H(t')] = 0$. The alternative is the Dyson series that converges absolutely whenever $t > 0$ and

bounded $||H(t)||$:

$$U(t) = \mathbf{1} - i\int_0^t H(t_1)dt_1 - \int_{t_2}^t\int_0^{t_2} H(t_2)H(t_1)dt_1dt_2 + i\int_{t_3}^t\int_{t_2}^{t_3}\int_0^{t_2} H(t_3)H(t_2)H(t_1)dt_1dt_2dt_3 + ... \tag{127}$$

We can rewrite the previous expression using the time ordering operator

$$U(t) = \mathcal{T}[e^{-i\int_0^t H(s)ds}] = \sum_{k=0}^{\infty}(-i)^k D_k. \qquad D_k = \frac{1}{k!}\int_0^t ...\int_0^t \mathcal{T}[H(t_k)...H(t_1)]d^kt. \tag{128}$$

As we did for the Taylor series, we have to truncate the series to order K such that the error remains lower than target ϵ_{HS}. We will see that K will be logarithmic in the corresponding precision.

Let us now focus on the input model. We need two definitions. The first is the usual block encoding

$$\text{HAM} = \begin{pmatrix} H/\lambda & \cdot \\ \cdot & \cdot \end{pmatrix} \Rightarrow (\langle 0|_a \otimes \mathbf{1}_s)\text{HAM}(|0\rangle_a \otimes \mathbf{1}_s) = \frac{H}{\lambda} \tag{129}$$

where we decompose HAM as in previous appendices

$$HAM = (\text{Prepare}^\dagger \otimes \mathbf{1}_s)\text{Select}(\text{Prepare} \otimes \mathbf{1}_s) \tag{130}$$

For a time dependent Hamiltonian we similarly define HAM-T as substituting H in the matrix form of HAM in (129) with

$$H = \text{Diagonal}[H(0), H(t/M), ..., H(1 - t/M)]. \tag{131}$$

In other words:

$$(\langle 0|_a \otimes \mathbf{1}_s)\text{HAM-T}(|0\rangle_a \otimes \mathbf{1}_s) = \sum_{l=0}^{M-1} |m\rangle\langle m| \otimes \frac{H\left(\frac{mt}{M}\right)}{\lambda}. \tag{132}$$

Having defined the main constructions for our algorithm, HAM and HAM-T, we now need the main theorem for simulating a time-dependent Hamiltonian for a short time segment:

Theorem 1. *[44] Let $H(s)$ be a time-dependent Hamiltonian such that $\max_s ||H(s)|| \leq \lambda$ and $\langle||\dot{H}||\rangle$ the average value of its time derivative. Let $M \in O\left(\frac{t^2}{\epsilon_{HS}}(\langle||\dot{H}||\rangle + \max_s ||H(s)||^2)\right)$. Then, for all $t \in [0, \frac{1}{2\lambda}]$ and $\epsilon_{HS} > 0$, exists W such that $||W - \mathcal{T}[e^{-i\int_0^t H(s)ds}]|| \leq \epsilon_{HS}$ with probability $1 - O(\epsilon_{HS})$, and $K = O\left(\frac{\log \epsilon_{HS}^{-1}}{\log\log \epsilon_{HS}^{-1}}\right)$ queries to HAM-T.*

The proof is given in Appendix B [44]. The key idea is that we want to approximate the time evolution operator with $W = \text{TDS}$, the oblivious amplitude amplification of $\text{TDS}_\beta = \sum_{k=0}^K \frac{(-it)^k}{M^k\beta} B_k$. As customary to require a single step of oblivious amplitude amplification, one takes $\beta = 2$.

The general strategy for TDS_β is similar to the Prepare Select Prepare† scheme. For the Select operator we first construct a sequence of K unitaries $U_1...U_K$ block-encoding matrices $H_1...H_K$:

$$(\langle 0|_a \otimes \mathbf{1}_s)U_k(|0\rangle_a \otimes \mathbf{1}_s) = H_k; \quad ||H_k|| \leq 1. \tag{133}$$

The consecutive applications of such matrices, $H_k...H_1 \propto B_k$, the k-th term in the Dyson series. DYS_K is the Select-like unitary that will apply this sequence $U_1...U_k$ controlled on index $|k\rangle$,

$$(\langle 0| \otimes \mathbf{1})DYS_K(|0\rangle \otimes \mathbf{1}) = \sum_{k=0}^K |k\rangle\langle k| \otimes \gamma_k B_K, \tag{134}$$

where $\gamma_k = M^{-k}$ will be a weighting coefficient of the Dyson series. Constructing such U_k operators is explained in the appendix B [44].

The second ingredient needed is Prepare-like operators $COEF$ and $COEF'^\dagger$, the difference between them being the phase in the Dyson series term. This allows us to perform the TDS_β operator (see fig. 6). Uniform_M, needed in the implementation of DYS_K can be implemented as suggested in the main reference for this appendix [5], while the rest are arithmetic operations, and the implementation of HAM-T discussed later on.

To extend Theorem 1 to longer time periods one can just apply it multiple times with the corresponding scaled error, as given by Corollary 4 of [44]. Since Theorem 1 indicates that the maximum simulation time for a single segment is $\tau = t/\lceil 2\lambda t\rceil$ with $\max_s ||H(s)|| \leq \lambda$, the number of segments is $r = \lceil 2\lambda t\rceil$, and the error allowed for each segment $\delta = \epsilon_{HS}/r$. Then Lemma 5 in [44] states that the constants K and M for the simulation of a single segment to error δ are

$$K = \left\lceil -1 + \frac{2\log(2r/\epsilon_{HS})}{\log\log(2r/\epsilon_{HS}) + 1}\right\rceil \tag{135a}$$

and

$$M = \left\{\frac{16\tau^2}{\delta}(\langle||\dot{H}||\rangle + \max_s ||H(s)||^2), K^2\right\}. \tag{135b}$$

The next step is to use this framework to simulate time-independent Hamiltonians in the interaction picture

$$H_I(t) = e^{iH_{S,0}t}H_{S,1}e^{-iH_{S,0}t}. \tag{136}$$

The advantage of simulating in this frame will happen when the norm of $||H_I(t)|| = ||H_{S,1}|| \ll ||H_{S,1}|| + ||H_{S,0}||$. We can apply this formalism to the Hamiltonian in the dual wave basis, equation (68), where $H_{S,0} = U + V$ and $H_{S,1} = T$.

To simulate our time-independent Hamiltonian we use the Theorem 1 and Corollary 4 in [44]. As

$$\begin{aligned}|\psi_S(t)\rangle &= e^{-iH_{S,0}t}\,|\psi_I(t)\rangle \\ &= e^{-iH_{S,0}t}\mathcal{T}[e^{-i\int_0^t H_I(s)ds}]\,|\psi(0)\rangle\end{aligned} \tag{137}$$

we can divide the evolution in r segments, $\tau = t/r$,

$$\begin{aligned}|\psi(t)\rangle &= (e^{-i(H_{S,0}+H_{S,1})\tau})^r\,|\psi(0)\rangle \\ &= \left(e^{-iH_{S,0}\tau}\mathcal{T}[e^{-i\int_0^\tau H_I(s)ds}]\right)^r|\psi(0)\rangle\,,\end{aligned} \tag{138}$$

and simulate it in a similar way as suggested by Corollary 4 in [44]. We have already explained ow to perform $\mathcal{T}[e^{-i\int_0^\tau H_I(s)ds}]$. However, we have yet to specify how implement HAM-T, which can be done as

$$\begin{aligned}\text{HAM-T} = &\left(\sum_{m=0}^{M-1}|m\rangle\langle m|_d\otimes\mathbf{1}_a\otimes e^{iH_{S,0}\tau m/M}\right)\cdot \\ \cdot(\mathbf{1}_d\otimes H_{S,1})\cdot&\left(\sum_{m=0}^{M-1}|m\rangle\langle m|_d\otimes\mathbf{1}_a\otimes e^{-iH_{S,0}\tau m/M}\right).\end{aligned} \tag{139}$$

We also have to explain how to implement $e^{-i(U+V)t}$. Notice that U and V commute because they are diagonal in the dual wave basis and can therefore be fast-forwarded. However, to avoid the $O(N^2)$ cost in the V term we will

1. Define the Fourier transform of V coefficients, $\tilde{V}(\vec{k}) = \sum_{\vec{x}} V(\vec{x})e^{2\pi i\vec{x}\cdot\vec{k}/N^{1/d}}$, and of the operators, $\tilde{\chi}_{\vec{k}} = \frac{1}{\sqrt{N}}\sum_{\vec{x}} e^{-2\pi i\vec{x}\cdot\vec{k}/N}\sum_\sigma n_{\vec{x},\sigma}$; and assuming V is real and symmetric, equation 39 from [44] writes

$$\begin{aligned}V &= \sum_{(\vec{x},\sigma)\neq(\vec{y},\sigma')} V(\vec{x}-\vec{y})n_{\vec{x},\sigma}n_{\vec{y},\sigma'} \\ &= \sum_{\vec{k}}\tilde{V}(\vec{k})\tilde{\chi}_{\vec{k}}\tilde{\chi}_{\vec{k}}^\dagger + \sum_{\vec{p},\sigma}\left(\sum_{\vec{k}}\tilde{V}(\vec{k})\right)n_{\vec{p},\sigma}.\end{aligned} \tag{140}$$

2. Use a binary oracle O_A such that $O_A\,|j\rangle\,|0\rangle_o\,|0\rangle_{garb} = |j\rangle\,|A_j\rangle_o\,|g(j)\rangle_{garb}$. Then we can implement the phase operator

$$\begin{aligned}&|j\rangle\,|0\rangle_o\,|0\rangle_{garb} \to_{\tilde{O}_A} |j\rangle\,|A_j\rangle_o\,|g(j)\rangle_{garb}\,|0\rangle \to_{PHASE} \\ &e^{-iA_jt}\,|j\rangle\,|A_j\rangle_o\,|g(j)\rangle_{garb}\,|0\rangle \to_{\tilde{O}_A^\dagger} e^{-iA_jt}\,|j\rangle\,|0\rangle_o\,|0\rangle_{garb}\end{aligned} \tag{141}$$

We want to implement the oracle O_V to calculate the Fourier Transform of V (omitting garbage registers), so this oracle can be decomposed as

$$\begin{aligned}\left(\bigotimes_{x,\sigma}|n_{x,\sigma}\rangle\right)|0\rangle &\to_{ADD}\bigotimes_x|\sum_\sigma n_{x,\sigma}\rangle \to_{FFT}\bigotimes_k|\tilde{\chi}_k\rangle \\ &\to_{|\cdot|^2}\bigotimes_k||\tilde{\chi}_k|^2\rangle \to_{\times V_k}\bigotimes_k|V_k|\tilde{\chi}_k|^2\rangle\,.\end{aligned} \tag{142}$$

Notice that $|n_x\rangle$ indicates the occupancy of the corresponding orbital, and as we are working with fermions, the FFT is the Fermionic Fast Fourier Transform.

H.2 How to compute its cost

To be able to count the complexity of the circuit it is useful to first indicate the size of each of the registers that appear in the algorithm, and more in particular in the analysis of the TDS operator in appendix B [44]:

1. Register s is the register containing the state. In second quantization it has size N.
2. Register a has a size given by the block encoding. Using [45] this can be bound by the logarithm of the number of unitary terms in Hamiltonian that are summed, $n_a = \lceil\log_2\Gamma\rceil$.
3. Register b has $n_b = \log_2(K+1)$ qubits.
4. Register c has $n_c = 1+\log_2(K+1)$ qubits.
5. Registers d and e require $\log_2 M$ qubits.
6. Register f only requires 1 qubit.

Secondly, we have to specify the value of K and M in (135). For K we already mentioned that $\delta = \epsilon_{HS}/r$, while in the usual definition of r we will take $\lambda = \lambda_1 = ||H_{S,1}|| = ||T||$. Instead of taking $\tau = \frac{1}{2\lambda_1}$ [44], we may take it slightly higher, $\tau = \frac{\ln 2}{\lambda_1}$ as this limit comes from the oblivious amplitude amplification technique [9], and we will do so to carry out similar treatment between the algorithms. Then, since $t = \frac{\pi}{\epsilon_{QPE}}$ this implies that $r := t/\tau = \lceil||T||t\rceil = \left\lceil\frac{\pi||T||}{\epsilon_{QPE}\ln 2}\right\rceil$.

Additionally, we need to obtain the value of M. Since $\max_s||H_I(s)|| \leq ||H_{S,1}||$ and $\langle||\dot{H}||\rangle = |||[H_{S,0},H_{S,1}]|| \leq 2||H_{S,0}||\cdot||H_{S,1}||$, substituting $\tau = \ln 2/\lambda_1$ and $\delta = \epsilon_{HS}/r = \epsilon_{HS}t/\tau$ in the value of M

$$\begin{aligned}M &= \max\left\{\frac{16\tau^2}{\delta}(\langle||\dot{H}||\rangle + \max_s||H(s)||^2), K^2\right\} \\ &= \max\left\{\frac{16t\ln 2}{\lambda_1\epsilon_{HS}}(2||H_{S,1}||||H_{S,0}|| + ||H_{S,1}||^2), K^2\right\} \\ &= \max\left\{\frac{16t\ln 2}{\epsilon_{HS}}(2||H_{S,0}|| + ||H_{S,1}||), K^2\right\}.\end{aligned} \tag{143}$$

To finish giving a description of the algorithm we need to particularize HAM-T for the time-independent Hamiltonian that we want to use, as

given in Lemma 7 in [44]

$$\text{HAM-T} = \left(\sum_{m=0}^{M-1} |m\rangle\langle m| \otimes \mathbf{1}_a \otimes e^{i(U+V)\tau m/M}\right) \cdot \\ \cdot (\mathbf{1} \otimes O_T) \cdot \left(\sum_{m=0}^{M-1} |m\rangle\langle m| \otimes \mathbf{1}_a \otimes e^{-i(U+V)\tau m/M}\right). \tag{144}$$

Here,

$$\left(\sum_{m=0}^{M-1} |m\rangle\langle m| \otimes \mathbf{1}_a \otimes e^{i(U+V)\tau m/M}\right)$$

can be implemented with $\lceil \log_2 M \rceil$ controlled-rotations of the kind $e^{i(U+V)\tau/M}$, $e^{i(U+V)\tau 2/M}$, $e^{i(U+V)\tau 4/M}$...

Lastly, performing O_T can be done using $O_T = (\text{Prepare}_T^\dagger \otimes \text{FFFT}^\dagger)\text{Select}_T(\text{Prepare}_T \otimes \text{FFFT})$, where

$$\text{Prepare}_T |0_a\rangle = \sum_p \sqrt{\frac{\tilde{T}(p)}{\lambda_T}} |\vec{p}\rangle, \tag{145a}$$

$$\text{Select}_T = \sum_p |\vec{p}\rangle\langle\vec{p}| \otimes n_p, \tag{145b}$$

and FFFT applied using $O(N \log N)$ gates, as we already discussed in appendix E.

Finally, let us highlight that there is a way to avoid the extra cost posed by Amplitude Amplification. The key idea is to implement a block encoding of $\sin(H\tau) = \frac{e^{+iH\tau} - e^{-iH\tau}}{2i}$, so that the qubitization walk operator will implement $e^{-i \arcsin \mu_j/\lambda'} = e^{-i \arcsin(\sin E_j\tau)/\exp(\lambda_1\tau)} \approx e^{-iE_j\tau/\exp(\lambda_1\tau)}$ [62], for $\mu_j = \sin E_j\tau$ and $\lambda' = \exp(\lambda_1\tau)$ [34]. If the segment time length of the amplitude amplified algorithm is $\tau \approx \ln 2/\lambda_1$, then the adjusted segment length is $\tau_{\text{eff}} \approx \ln 2/2\lambda_1$, and if one increases $\tau \approx 1/\lambda_1$, then $\tau_{\text{eff}} \approx 1/(e\lambda_1)$. This means that each step of the algorithm does not need to be amplified, but the number of time segments increases from $\lambda_1/(\epsilon_{QPE} \ln 2)$ to $e\lambda_1/\epsilon_{QPE}$ [62]. In this case we also aim to perform Hamiltonian simulation over $H - \tilde{E}_0$, where $\tilde{E}_0$ is an approximation to the ground state energy, so that we operate on the linear regime of the sine and arcsine functions, and the error introduced is small.

H.3 How to adapt the Hamiltonian simulation to control the direction of the time evolution.

We will use the trick of setting the controlled evolution of phase estimation $|1\rangle|\phi\rangle \to e^{i\phi}|1\rangle|\phi\rangle$ and $|0\rangle|\phi\rangle \to e^{-i\phi}|0\rangle|\phi\rangle$. This will be reflected in the Dyson expansion, where we will have to add a i factor to the coefficients in $COEF$ conditional on the phase estimation ancilla being on state 1; and in the sign of the exponential $e^{-i(U+V)\tau}$.

9

Applications to Electric Batteries Design

The electric battery industry includes any process related to the manufacture, marketing, use or repair of a battery. This chapter is especially focused on manufacturing. A key process in the manufacture of an electric battery is the selection and design of its materials. For this, simulation of the behavior of materials is a key process and can be achieved using quantum chemistry algorithms.

Problems such as the design of stronger, lighter, or cheaper materials, the chemical reaction to achieve propulsion, or the architecture of new electric batteries with better properties are concrete examples of the challenges faced by manufacturing companies. These are problems of enormous complexity in which the most advanced computational techniques must be used to obtain results. For this reason, these companies always look to the research field with great interest in the new techniques it can offer them. These techniques include high-performance computing, deep learning, and quantum computing.

One of the main questions companies in the electric battery sector, as in other sectors, are asking themselves is when the technology needed to run quantum computing algorithms will become available. The interest is twofold, first because they want to make breakthroughs of their own, second because no company wants to lag behind its competitors' advances. At this point, TFermion can become a fundamental tool because it allows estimating the quantum hardware resources needed to run high-impact algorithms in battery design problems.

In many cases, for companies, it is not so important what is created, but rather when it is created and what is needed to create it. In order to get ahead of the future era of quantum computing battery design, companies are interested in sizing the amount of resources that

running quantum algorithms can cost them. This can be used to make strategic decisions in the medium term, based on whether or not quantum algorithms will exist with practical advantage, deciding whether to purchase a quantum computer, depending on the amount of resources needed, it may be more efficient to outsource a quantum computer to the cloud, etc.

9.1 Introduction to electric batteries design

The current situation in electric battery manufacturing is especially changing with the arrival of many electric devices, as laptops, smartphones, autonomous vehicles, etc. For example, an electric car is a major paradigm shift, not only for the car itself, but also for its needs. For example, if the vehicle fleet is mostly electric, gas stations will see their business dwindle and will have to convert to charging points. Therefore, any tool that provides information on the timescale of this change is very useful.

The optimization of lithium-ion battery performance is pivotal for the advancement of next-generation energy storage systems. Progress in this field relies not only on the discovery of novel materials but also on the refinement of simulation methods to accurately characterize key properties of lithium-ion batteries. The spectrum of properties influencing their performance is diverse and encompasses mechanical and electrochemical attributes, thermal stability of the cathode, the electrochemical windows of the electrolyte, formation of the solid-electrolyte interphase, and ionic mobility, among others. This chapter is based on publication [1].

Today, the limiting factor in the deployment of electric vehicles is their range, that is, the number of kilometers they are able to travel without recharging the battery. Although this measure depends on numerous factors, such as aerodynamics, vehicle weight, driving style, tires, etc., the factor with the greatest impact is the capacity of the electric batteries. These batteries have been increasing their capacity to have autonomies that range from 400 to 800 km. However, it is expected that this range will be much greater with the development of technology.

To increase the capacity of an electric battery, it must be possible to simulate the chemical processes that occur inside the battery. Understanding the optimum structure of the anode and cathode, how to achieve the best storage and discharge rate, or how to avoid battery degradation are fundamental processes for extending the autonomy and life of these batteries.

All these processes have in common the need to use a GSP and GSEE algorithm. As seen above, the classical algorithms that do these tasks have low accuracy due to their limited scalability. Therefore, quantum algorithms have become the most promising solution in the medium term. There are numerous approaches to accelerate the development of new algorithms, TFermion proposes to study their cost and compare the techniques that implement each of them to obtain certain conclusions.

Following this motivation, this paper reports the collaboration with the start-up Xanadu and the company Volkswagen for the application of TFermion to the study of quantum algorithms that allow to know and design with detail and precision electric batteries.

9.2 Methodology

The task of simulating all the processes that occur in an electric battery is too complex, not to mention that there are numerous technologies for creating electric batteries. In this case, the work is focused on the simulation of a lithium-ion battery, and the study focused on the cathode.

The unit cell material, Li2FeSiO4, is studied because it is a common candidate material for the cathode of batteries. Other properties such as the ionic mobility inside the cathode, and the thermal stability of the material, can also be predicted from the ground state energy of the different phases of the material. In summary, this work is focused on the simulation of cathode materials, which is crucial for predicting important properties of a battery cell.

The battery cell comprises a positive electrode (cathode) and a negative electrode (anode), separated by a porous membrane (separator) and immersed in an ion-conducting material (electrolyte). Chemical reactions occurring at the electrode-electrolyte interface drive the conversion of chemical energy into electrical energy. During discharge, an oxidation reaction at the anode generates electrons and lithium ions. The electrons flow through an external circuit, while the lithium ions migrate through the electrolyte and intercalate into the cathode material. Conversely, during charging, an external voltage is applied to reverse this process. The lithium ions are extracted from the cathode, transported back through the electrolyte, and intercalated into the anode material.

In this chapter, three fundamental aspects of electric batteries are investigated: the equilibrium cell voltage, the ionic mobility, and the thermal stability of the cathode material. These aspects are critical for enhancing the capacity of lithium-ion batteries and are instrumental in studying the electronic structure of cathode materials.

The equilibrium voltage plays a crucial role in defining the energy storage capacity of a battery relative to its volume (energy density) and weight (specific energy). It represents the average voltage (V) of an electrochemical cell, which converts chemical energy into electrical energy. The Nernst equation governs the equilibrium voltage, taking into account factors such as the number of charges transferred, the Faraday constant, and the change in free energy associated with the cell reaction [131].

The chemical diffusivity (D) serves as a key parameter characterizing the mobility of lithium

ions within a material. In scenarios where diffusion mechanisms remain independent of temperature, a microscopic model can effectively illustrate the process of lithium ion hopping from its initial position to an adjacent vacant site within the crystal structure of the host material.

Numerous processes contribute to the degradation of battery performance over time, encompassing the formation of the solid electrolyte interphase, degradation of cathode active materials, lithium plating on the anode, and the growth of lithium dendrites, among others [132]. Simulating these phenomena poses a significant challenge, as it requires a bottom-up approach spanning from the atomic level to the macroscopic scale [133].

Anticipating the thermal stability of cathode materials is crucial for enhancing the safety of lithium-ion batteries, particularly as they can become unstable in their charged state. The removal of more lithium ions from oxide-based cathode materials can lead to their degradation into other phases of the material [134].

At present, first-principles calculations of the electronic structure of cathode materials are largely performed using density functional theory (DFT) methods [135], explained in chapter 8. This chapter is dedicated to exploring methods suitable for calculating the ground-state energies of cathode materials, a crucial parameter for simulating battery properties, specifically quantum phase estimation algorithm. A quantum phase estimation (QPE) algorithm has been used with simulation of the Hamiltonian by qubitization, plane waves to represent the quantum state, and first quantization to encode the system.

The quantum algorithm begins by taking the Hamiltonian, which describes the interactions among electrons within the material's unit cell, as input. It then generates an approximation of the smallest eigenvalue of this Hamiltonian, which corresponds to the ground-state energy. To facilitate this process, a method for representing and constructing Hamiltonians is necessary, tailored specifically to the quantum algorithm.

Studying the electronic Hamiltonian is crucial to understand why employing a first-quantization approach in a plane-wave basis is well-suited for simulating battery materials [126]. This approach allows for an effective representation of the electronic structure and interactions within the material, making it particularly suitable for quantum simulations.

However, preparing an approximate ground state to serve as input for the quantum phase estimation algorithm presents challenges, especially in the case of periodic materials and within the context of first quantization. Careful consideration is required to identify suitable methods for achieving this task effectively.

Once the ground state is prepared, the qubitization formalism comes into play, enabling the encoding of the Hamiltonian into a suitable unitary transformation [23]. This step is essential for implementing the quantum phase estimation algorithm.

To analyze the overall complexity of the algorithm and ensure compatibility with fault-tolerant architectures, TFermion is employed. TFermion facilitates the compilation of all necessary operations into a universal set of quantum gates, ensuring the algorithm's feasibility and scalability on quantum hardware.

Plane waves offer a well-suited framework for studying periodic systems, providing concise representations of Hamiltonians. However, achieving high accuracy often demands a significant number of plane waves, resulting in an impractically large number of qubits in second quantization. To address this challenge, first-quantization techniques are favored for materials simulation. This approach leverages qubitization-based quantum phase estimation algorithms, specifically designed for first-quantized Hamiltonians represented in a plane-wave basis.

The atomic configuration of a cathode material is characterized by its unit cell, comprising a collection of atoms that can be spatially translated to cover the entire crystal. To determine the material's electronic structure, the Schrödinger equation is solved within the unit cell, subject to periodic boundary conditions. While any comprehensive set of basis functions can represent the first-quantized Hamiltonian H, for periodic systems, employing plane waves with the lattice's periodicity emerges as a fitting selection.

In the quantum phase estimation algorithm, achieving an input state with substantial overlap with the actual ground state is crucial. In many quantum chemistry algorithms, including quantum phase estimation, this is accomplished by generating a state of non-interacting electrons characterized by single-particle wave functions, or orbitals, which are optimized using the Hartree-Fock method.

The situation is more complicated when studying periodic materials in first quantization using a plane-wave basis. Here it is necessary to apply the Hartree-Fock method to extended materials and provide an algorithm to prepare the resulting Hartree-Fock state in a plane-wave basis, which must be explicitly antisymmetrized.

Studying periodic materials in first quantization using a plane-wave basis presents a more intricate scenario. In this context, applying the Hartree-Fock method to extended materials becomes necessary. Moreover, an algorithm is required to prepare the resulting Hartree-Fock state in a plane-wave basis, with explicit antisymmetrization.

For the Quantum Phase Estimation (QPE) algorithm is necessary to select a method to encode the Hamiltonian into an appropriate unitary operator. The qubitization method stands out due to its ability to achieve exact implementation without any approximation. Quantum phase estimation, executed on the qubitization operator, relies on operators of preparation, $PREP_H$, and selection, SEL_H. The compilation strategy for implementing these operators is derived from [136].

Unlike the methods analyzed in TFermion, for this particular use case, the cost of preparing the ground state approach has been taken into account. This cost has been divided into cost in T-gates, Toffoli gates and an estimate of the algorithm execution time in quantum hardware.

To carry out this study, it has been necessary to make slight modifications to the TFermion library. First, the combination of techniques to be used for the simulation was included in a concrete way. In addition, new performance metrics, Toffoli gates, and execution time have been included. Finally, the code has been optimized to include a calculation of the cost of the algorithm for a molecule of a higher complexity than those analyzed in the TFermion work.

Figure 31 illustrates the relationship between the overall cost of the algorithm, quantified by the number of Toffoli gates, and varying parameters such as the number of plane waves (N) and the error in estimating the ground-state energy. Given the discrete nature of qubit requirements for representing the quantum state, which directly influences the Toffoli gate cost (set at 100), it is advisable to select a number of plane waves (N) corresponding to integer values of np.

The insights from figure 32 shed light on the computational demands of executing a quantum algorithm, particularly in relation to precision requirements and clock frequency. At a modest clock rate of 10 kHz, even with a low precision of np = 4, the estimated runtime extends over several years. Conversely, with an optimistic but feasible clock rate of 100 MHz, the runtime for np = 4 is reduced to less than a day. However, for np = 9, the estimated runtime extends to approximately a year. It may be prudent to limit the number of plane waves to $N = 10^6$, corresponding to np = 7, as this adequately captures the system's accuracy even in large basis sets like cc-pV5Z. With this choice and a 100 MHz clock rate, the runtime is projected to be approximately a few weeks.

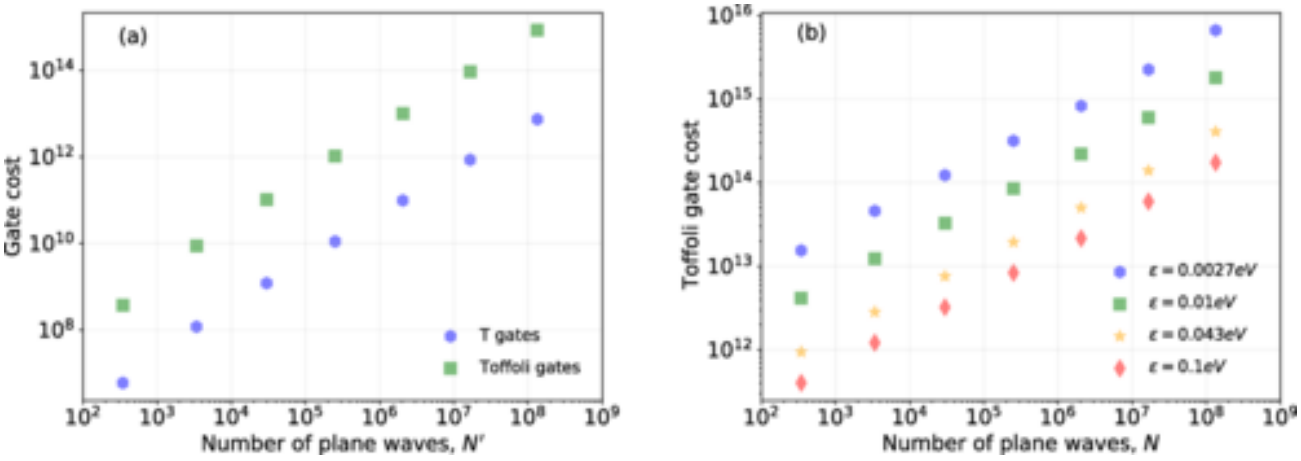

Figure 31: Non-Clifford gate cost for initial state preparation and quantum phase estimation. (a) The non-Clifford gate cost due to Givens rotations used in the circuit for initial state preparation. (b) Toffoli gate cost of the quantum phase estimation algorithm. All calculations are done for the unit cell of Li2FeSiO4 with 156 electrons. The total number of qubits is 2,375 for np = 4 and 6,652 for np = 9. In the right figure it only depicts Toffoli gate count, as the number of T gates is much smaller (¡ 3 × 105). The total error includes contributions from different approximations throughout the algorithm, but it does not take into account the error derived from a finite basis set. The slope of the Toffoli gate cost for fixed target precision is a consequence of the leading cost term. These calculations were performed with the T-Fermion library.

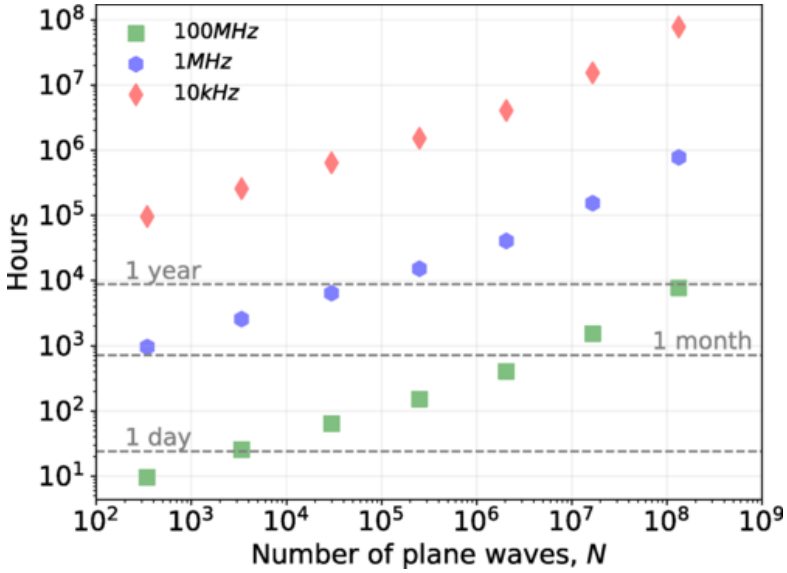

Figure 32: Estimation of the time required to run the algorithm. This figure illustrates total runtime for synthesizing all the Toffoli gates indicated in figure 31. All calculations are done for the unit cell of Li2FeSiO4 with 156 electrons, and it is assumed that the number of plane waves used in the state preparation and quantum phase estimation are the same. The total number of qubits is 2,375 for np = 4 and 6,652 for np = 9. The distillation time is computed as the product of the number of Toffoli gates, the surface code distance d, and the clock frequency, all divided by a small np factor that parallelize the CSWAPs and arithmetic computations. It is emphasized that these are rough estimates whose main purpose is to provide a method to interpret the gate cost.

9.3 Results

- ✓ Application of TFermion to a highly complex industrial problem, with results that help to learn more about the needs of quantum devices.
- ✓ Extension of the TFermion library to obtain two new metrics, Toffoli gates and execution time. In addition, the analysis of ground state preparation cost has been included.
- ✓ Analysis of how quantum chemistry algorithms should be improved to approach the capabilities of quantum hardware.
- ✓ Demonstration that the resources required to run Quantum Phase Estimation algorithms on quantum hardware and gain practical quantum advantage, while far from the capabilities of current devices, are in line with medium-term device growth estimates.

Conclusions

This thesis has focused on the study of algorithms that follow the hybrid classical-quantum paradigm. As has been explained, this paradigm represents a middle way that tries to take advantage of the widely studied classical algorithms and the new possibilities offered by quantum algorithms. Although it may not be a definitive solution, it seems more logical to move towards a scenario in which the algorithms are either classical or quantum, it is an intermediate step that facilitates the execution of quantum algorithms and possible scenarios in which existing algorithms are accelerated in highly complex problems.

Hybrid algorithms encompass a heterogeneous group of techniques ranging from classical algorithms with a quantum module in charge of performing a complex but bounded task, algorithms that only include a classical preprocessing module to reduce the volume of data to be encoded in the classical circuit, to architectures in which the quantum circuit is optimized by a classical algorithm. There are even other formulations in which a classical algorithm analyzes a quantum algorithm to optimize its cost and analyze its properties.

The first problems that quantum computing turned its attention to were simulation problems. Later, it was realized that the best scenario for quantum simulation with practical applications was quantum chemistry. In the next generation of quantum computing development and with more algorithms proposed, optimization problems also began to be thought of as good candidates to be accelerated by quantum algorithms. Therefore, in this work, these two families of problems, optimization and quantum chemistry, have been chosen to apply the developed algorithms. The motivation behind this choice was to create algorithms that can have a high impact on the industrial sector.

In relation to optimization problems, the study has focused on a specific subcategory, namely search & sample algorithms. This subcategory identifies algorithms capable of performing a search in a state space to find the distribution of the data, or a minimum or a maximum. For this purpose, the main classical algorithms belonging to this category have been analyzed, and their limitations have been identified.

Different quantum proposals and their limitations were analyzed. Then, one has chosen the one that has until now had aroused less interest in the world of quantum optimization, the quantum walks. Using quantum walks as the central algorithm, other authors have built on

them a quantum version of the Metropolis-Hastings algorithm. In this work, these versions of the M-H quantum algorithm were taken as a reference and modified to implement it in a quantum circuit. This implementation has resulted in a software tool called quantum Metropolis Solver, QMS, which has been applied to different use cases.

QMS has been applied to the artificial intelligence use case because the fundamental process of many AI algorithms is search, especially machine learning algorithms. ML algorithms perform an abstraction and inference process, in which they require a search process to find the best hypothesis to explain the input examples. Since performing tests on these algorithms is too complex due to the limited capabilities of quantum hardware, a reduced problem that is classically used as a benchmark for these problems, the N-Queen problem, has been used. The results show that there is a polynomial advantage of the hybrid algorithm over the purely quantum algorithm.

Another field in which there are numerous problems that can be understood in the context of optimization and where classical algorithms have strong scalability limitations is the analysis of space exploration data. In this use case, this work focused on the analysis of gravitational waves. For this purpose, QMS was adapted to the gravitational wave parameter estimation problem, observing quantum advantage over a classical algorithm used by the LIGO collaboration to obtain its results.

The last use case in which QMS was applied is the protein folding problem. In this case, there are many processes in nature that can only be simulated by trying many combinations until the one that best fits the description of the problem is found. In this case, QMS was applied to protein folding as an assistant to a classical deep learning algorithm. Results were obtained that validated the initial intuition that a quantum search algorithm was able to improve the initial results offered by a deep learning algorithm.

In the quantum chemistry part II, a different methodology for hybrid algorithms has been presented. In this case, the hybrid algorithm allows a detailed analysis of the cost of quantum algorithms when executed on a quantum device.

For this purpose, a software tool, TFermion, has been designed to calculate the T-gate cost of a quantum algorithm applying Quantum Phase Estimation. The results have made it possible to understand and compare the cost of state-of-the-art quantum chemistry algorithms and the evolution required by quantum devices to run these algorithms.

Finally, the application of TFermion on a real problem with industrial interest, the design of electric batteries, has allowed to understand the usefulness of this tool. The results of TFermion in this problem have shown that the algorithms based on simulation of the Hamiltonian by qubitization, plane waves to represent the quantum state and first quantization to encode the

system, obtain the best results for the design of the cathode.

This thesis aims to demonstrate that hybrid algorithms offer, at present, many advantages for the main problems that are candidates for the early fault-tolerant quantum advantage. In addition, this hybrid paradigm enables the implementation of the algorithms in current and future devices in the near term by reducing the size and depth of quantum circuits. Hybrid algorithms do not seem to be the ultimate solution but they are a small step for algorithm design but a big step for the quantum computing giant.

References

[1] A. Delgado, P. A. Casares, R. Dos Reis, M. S. Zini, R. Campos, N. Cruz-Hernández, A.-C. Voigt, A. Lowe, S. Jahangiri, M. A. Martin-Delgado, *et al.*, "Simulating key properties of lithium-ion batteries with a fault-tolerant quantum computer," *Physical Review A*, vol. 106, no. 3, p. 032428, 2022.

[2] G. Escrig, R. Campos, P. A. Casares, and M. Martin-Delgado, "Parameter estimation of gravitational waves with a quantum metropolis algorithm," *Classical and Quantum Gravity*, vol. 40, no. 4, p. 045001, 2023.

[3] G. Escrig, R. Campos, H. Qi, and M. Martin-Delgado, "Quantum bayesian inference with renormalization for gravitational waves," *arXiv preprint arXiv:2403.00846*, 2024.

[4] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985.

[5] A. M. Turing *et al.*, "On computable numbers, with an application to the entscheidungsproblem," *J. of Math*, vol. 58, no. 345-363, p. 5, 1936.

[6] R. S. Sutor, *Dancing with Qubits: How quantum computing works and how it can change the world*. Packt Publishing Ltd, 2019.

[7] A. Giani and Z. Eldredge, "Quantum computing opportunities in renewable energy," *SN Computer Science*, vol. 2, no. 5, p. 393, 2021.

[8] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[9] S. Aaronson, "Bqp and the polynomial hierarchy," in *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 141–150, 2010.

[10] J. M. Chow, J. M. Gambetta, A. D. Corcoles, S. T. Merkel, J. A. Smolin, C. Rigetti, S. Poletto, G. A. Keefe, M. B. Rothwell, J. R. Rozen, *et al.*, "Universal quantum gate set approaching fault-tolerant thresholds with superconducting qubits," *Physical review letters*, vol. 109, no. 6, p. 060501, 2012.

[11] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, no. 1907, pp. 553–558, 1992.

[12] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 6, pp. 818–830, 2013.

[13] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, 1996.

[14] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, pp. 124–134, Ieee, 1994.

[15] O. Kodheli, E. Lagunas, N. Maturo, S. K. Sharma, B. Shankar, J. F. M. Montoya, J. C. M. Duncan, D. Spano, S. Chatzinotas, S. Kisseleff, *et al.*, "Satellite communications in the new space era: A survey and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 70–109, 2020.

[16] P. A. M. Casares, R. Campos, and M. A. Martin-Delgado, "Qfold: quantum walks and deep learning to solve protein folding," *Quantum Science and Technology*, vol. 7, no. 2, p. 025013, 2022.

[17] C. Severance and K. Dowd, *High performance computing*. OpenStax CNX, 2010.

[18] T. S. Humble, A. McCaskey, D. I. Lyakh, M. Gowrishankar, A. Frisch, and T. Monz, "Quantum computers for high-performance computing," *IEEE Micro*, vol. 41, no. 5, pp. 15–23, 2021.

[19] S. Endo, Z. Cai, S. C. Benjamin, and X. Yuan, "Hybrid quantum-classical algorithms and quantum error mitigation," *Journal of the Physical Society of Japan*, vol. 90, no. 3, p. 032001, 2021.

[20] K. A. Britt and T. S. Humble, "High-performance computing with quantum processing units," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, pp. 1–13, 2017.

[21] Nvidia, "What is a qpu?." `https://blogs.nvidia.com/blog/what-is-a-qpu/`, 2022. [Online; accessed January 2024].

[22] S. Giordano and M. A. Martin-Delgado, "Reinforcement-learning generation of four-qubit entangled states," *Physical Review Research*, vol. 4, no. 4, p. 043056, 2022.

[23] G. H. Low and I. L. Chuang, "Hamiltonian simulation by qubitization," *Quantum*, vol. 3, p. 163, 2019.

[24] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[25] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[26] Y. Wu, W.-S. Bao, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan, *et al.*, "Strong quantum computational advantage using a superconducting quantum processor," *Physical review letters*, vol. 127, no. 18, p. 180501, 2021.

[27] L. S. Madsen, F. Laudenbach, M. F. Askarani, F. Rortais, T. Vincent, J. F. Bulmer, F. M. Miatto, L. Neuhaus, L. G. Helt, M. J. Collins, *et al.*, "Quantum computational advantage with a programmable photonic processor," *Nature*, vol. 606, no. 7912, pp. 75–81, 2022.

[28] P. Andreasson, J. Johansson, S. Liljestrand, and M. Granath, "Quantum error correction for the toric code using deep reinforcement learning," *Quantum*, vol. 3, p. 183, 2019.

[29] E. K. Donald *et al.*, "The art of computer programming," *Sorting and searching*, vol. 3, no. 426-458, p. 4, 1999.

[30] S. J. Russell and P. Norvig, *Artificial intelligence a modern approach*. London, 2010.

[31] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[32] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *International work-conference on artificial neural networks*, pp. 758–770, Springer, 2005.

[33] S. Aaronson, "The equivalence of sampling and searching," *Theory of Computing Systems*, vol. 55, no. 2, pp. 281–298, 2014.

[34] M. W. Krentel, "The complexity of optimization problems," in *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pp. 69–76, 1986.

[35] J. Maltese, B. M. Ombuki-Berman, and A. P. Engelbrecht, "A scalability study of many-objective optimization algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 79–96, 2016.

[36] A. Ambainis, "Quantum search algorithms," *ACM SIGACT News*, vol. 35, no. 2, pp. 22–35, 2004.

[37] A. Galindo and M. A. Martin-Delgado, "Family of grover's quantum-searching algorithms," *Physical Review A*, vol. 62, no. 6, p. 062303, 2000.

[38] M. Jünger, G. Reinelt, and G. Rinaldi, "The traveling salesman problem," *Handbooks in operations research and management science*, vol. 7, pp. 225–330, 1995.

[39] K. L. Hoffman, M. Padberg, G. Rinaldi, *et al.*, "Traveling salesman problem," *Encyclopedia of operations research and management science*, vol. 1, pp. 1573–1578, 2013.

[40] H. M. Salkin and C. A. De Kluyver, "The knapsack problem: a survey," *Naval Research Logistics Quarterly*, vol. 22, no. 1, pp. 127–144, 1975.

[41] K. M. Bretthauer and B. Shetty, "The nonlinear knapsack problem–algorithms and applications," *European Journal of Operational Research*, vol. 138, no. 3, pp. 459–472, 2002.

[42] C. Bowtell and P. Keevash, "The n-queens problem," *arXiv preprint arXiv:2109.08083*, 2021.

[43] D. P. Kroese and R. Y. Rubinstein, "Monte carlo methods," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 1, pp. 48–58, 2012.

[44] J. L. Villegas, E. Castro, J. Gutiérrez, *et al.*, "Representations in problem solving: A case study with optimization problems," *Electronic Journal of Research in Educational Psychology*, 2009.

[45] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo, *Limits to parallel computation: P-completeness theory*. Oxford University Press, USA, 1995.

[46] S. A. Cook, "Towards a complexity theory of synchronous parallel computation," in *Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook*, pp. 219–244, ACM Books, 2023.

[47] A. S. Singh and M. B. Masuku, "Sampling techniques & determination of sample size in applied statistics research: An overview," *International Journal of economics, commerce and management*, vol. 2, no. 11, pp. 1–22, 2014.

[48] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[49] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[50] O. Goldreich, "Computational complexity: a conceptual perspective," *ACM Sigact News*, vol. 39, no. 3, pp. 35–39, 2008.

[51] A. Wigderson, "P, np and mathematics–a computational complexity perspective," in *Proceedings of the ICM*, vol. 6, pp. 665–712, 2006.

[52] P. Beame, S. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi, "The relative complexity of np search problems," in *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pp. 303–314, 1995.

[53] C. Gambella, B. Ghaddar, and J. Naoum-Sawaya, "Optimization problems for machine learning: A survey," *European Journal of Operational Research*, vol. 290, no. 3, pp. 807–828, 2021.

[54] B. H. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial optimization*, vol. 1. Springer, 2011.

[55] J. Nievergelt, "Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power," in *International Conference on Current Trends in Theory and Practice of Computer Science*, pp. 18–35, Springer, 2000.

[56] F. Xia, J. Liu, H. Nie, Y. Fu, L. Wan, and X. Kong, "Random walks: A review of algorithms and applications," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 2, pp. 95–107, 2019.

[57] S. Richter, M. Helmert, and M. Westphal, "Landmarks revisited.," in *AAAI*, vol. 8, pp. 975–982, 2008.

[58] A. Candra, M. A. Budiman, and K. Hartanto, "Dijkstra's and a-star in finding the shortest path: A tutorial," in *2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA)*, pp. 28–32, IEEE, 2020.

[59] J. Hammersley, *Monte carlo methods*. Springer Science & Business Media, 2013.

[60] S. Ferson, "What monte carlo methods cannot do," *Human and Ecological Risk Assessment: An International Journal*, vol. 2, no. 4, pp. 990–1007, 1996.

[61] D. Milojicic, P. Faraboschi, N. Dube, and D. Roweth, "Future of hpc: Diversifying heterogeneity," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 276–281, IEEE, 2021.

[62] R. Babbush, J. R. McClean, M. Newman, C. Gidney, S. Boixo, and H. Neven, "Focus beyond quadratic speedups for error-corrected quantum advantage," *PRX Quantum*, vol. 2, no. 1, p. 010103, 2021.

[63] T. Patel, A. Wagenhäuser, C. Eibel, T. Hönig, T. Zeiser, and D. Tiwari, "What does power consumption behavior of hpc jobs reveal?: Demystifying, quantifying, and predicting power consumption characteristics," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 799–809, IEEE, 2020.

[64] C. Böhm, S. Berchtold, and D. A. Keim, "Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases," *ACM Computing Surveys (CSUR)*, vol. 33, no. 3, pp. 322–373, 2001.

[65] N. S. Yanofsky and M. A. Mannucci, *Quantum computing for computer scientists*. Cambridge University Press, 2008.

[66] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," *Contemporary Mathematics*, vol. 305, pp. 53–74, 2002.

[67] M. Szegedy, "Quantum speed-up of markov chain based algorithms," in *45th Annual IEEE symposium on foundations of computer science*, pp. 32–41, IEEE, 2004.

[68] P. A. M. Casares, "Fault-tolerant quantum algorithms," *arXiv preprint arXiv:2301.08057*, 2023.

[69] F. Magniez, A. Nayak, J. Roland, and M. Santha, "Search via quantum walk," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pp. 575–584, 2007.

[70] G. D. Paparo, V. Dunjko, A. Makmal, M. A. Martin-Delgado, and H. J. Briegel, "Quantum speedup for active learning agents," *Physical Review X*, vol. 4, no. 3, p. 031002, 2014.

[71] J. Lemieux, B. Heim, D. Poulin, K. Svore, and M. Troyer, "Efficient quantum walk circuits for metropolis-hastings algorithm," *Quantum*, vol. 4, p. 287, 2020.

[72] R. Campos, P. A. Casares, and M. Martin-Delgado, "Quantum metropolis solver: a quantum walks approach to optimization problems," *Quantum Machine Intelligence*, vol. 5, no. 2, p. 28, 2023.

[73] A. Montanaro, "Quantum walk speedup of backtracking algorithms," *arXiv preprint arXiv:1509.02374*, 2015.

[74] I. Verenich, H. Nguyen, M. La Rosa, and M. Dumas, "White-box prediction of process performance indicators via flow analysis," in *Proceedings of the 2017 International Conference on Software and System Process*, pp. 85–94, 2017.

[75] S. Morita and H. Nishimori, "Mathematical foundation of quantum annealing," *Journal of Mathematical Physics*, vol. 49, no. 12, 2008.

[76] P. Date, D. Arthur, and L. Pusey-Nazzaro, "Qubo formulations for training machine learning models," *Scientific reports*, vol. 11, no. 1, p. 10029, 2021.

[77] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, "Quantum annealing for industry applications: Introduction and review," *Reports on Progress in Physics*, 2022.

[78] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, "Perspectives of quantum annealing: Methods and implementations," *Reports on Progress in Physics*, vol. 83, no. 5, p. 054401, 2020.

[79] J. A. Miszczak, "Models of quantum computation and quantum programming languages," *arXiv preprint arXiv:1012.6035*, 2010.

[80] O. Di Matteo, V. Gheorghiu, and M. Mosca, "Fault-tolerant resource estimation of quantum random-access memories," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–13, 2020.

[81] K. L. Mengersen and R. L. Tweedie, "Rates of convergence of the hastings and metropolis algorithms," *The annals of Statistics*, vol. 24, no. 1, pp. 101–121, 1996.

[82] A. Lorberfeld, "Machine learning algorithms in layman's terms." `https://towardsdatascience.com/machine-learning-algorithms-in-laymans-terms-part-1-d0368d76` 2019. [Online; accessed January 2024].

[83] M. Chowdhury and A. W. Sadek, "Advantages and limitations of artificial intelligence," *Artificial intelligence applications to critical transportation issues*, vol. 6, no. 3, pp. 360–375, 2012.

[84] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for machine learning: Challenges and opportunities," in *2017 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–8, IEEE, 2017.

[85] M. M. Waldrop, "The chips are down for moore's law," *Nature News*, vol. 530, no. 7589, p. 144, 2016.

[86] G. Moore, "Moore's law," *Electronics Magazine*, vol. 38, no. 8, p. 114, 1965.

[87] G. E. Moore, "Lithography and the future of moore's law," in *Integrated Circuit Metrology, Inspection, and Process Control IX*, vol. 2439, pp. 2–17, SPIE, 1995.

[88] U. Gupta, Y. G. Kim, S. Lee, J. Tse, H.-H. S. Lee, G.-Y. Wei, D. Brooks, and C.-J. Wu, "Chasing carbon: The elusive environmental footprint of computing," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 854–867, IEEE, 2021.

[89] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.

[90] M. Cerezo, M. Larocca, D. García-Martín, N. Diaz, P. Braccia, E. Fontana, M. S. Rudolph, P. Bermejo, A. Ijaz, S. Thanasilp, *et al.*, "Does provable absence of barren plateaus imply classical simulability? or, why we need to rethink variational quantum computing," *arXiv preprint arXiv:2312.09121*, 2023.

[91] L. Bittel and M. Kliesch, "Training variational quantum algorithms is np-hard," *Physical review letters*, vol. 127, no. 12, p. 120502, 2021.

[92] R. Wiersema, D. Lewis, D. Wierichs, J. Carrasquilla, and N. Killoran, "Here comes the su(n): multivariate quantum gates and gradients," *arXiv preprint arXiv:2303.11355*, 2023.

[93] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature communications*, vol. 9, no. 1, p. 4812, 2018.

[94] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, "Connecting ansatz expressibility to gradient magnitudes and barren plateaus," *PRX Quantum*, vol. 3, no. 1, p. 010313, 2022.

[95] LANL, "Solving 'barren plateaus' is the key to quantum machine learning." `https://discover.lanl.gov/news/0319-barren-plateaus/`, 2021. [Online; accessed January 2024].

[96] A. Abbas, R. King, H.-Y. Huang, W. J. Huggins, R. Movassagh, D. Gilboa, and J. R. McClean, "On quantum backpropagation, information reuse, and cheating measurement collapse," *arXiv preprint arXiv:2305.13362*, 2023.

[97] K. Gili, M. Mauri, and A. Perdomo-Ortiz, "Evaluating generalization in classical and quantum generative models," *arXiv preprint arXiv:2201.08770*, 2022.

[98] M. S. Rudolph, S. Sim, A. Raza, M. Stechly, J. R. McClean, E. R. Anschuetz, L. Serrano, and A. Perdomo-Ortiz, "Orqviz: Visualizing high-dimensional landscapes in variational quantum algorithms," *arXiv preprint arXiv:2111.04695*, 2021.

[99] P.-L. Dallaire-Demers and N. Killoran, "Quantum generative adversarial networks," *Physical Review A*, vol. 98, no. 1, p. 012324, 2018.

[100] P. Casares and M. Martin-Delgado, "A quantum active learning algorithm for sampling against adversarial attacks," *New Journal of Physics*, vol. 22, no. 7, p. 073026, 2020.

[101] V. Leyton-Ortega, A. Perdomo-Ortiz, and O. Perdomo, "Robust implementation of generative modeling with parametrized quantum circuits," *Quantum Machine Intelligence*, vol. 3, no. 1, p. 17, 2021.

[102] S. Lloyd and C. Weedbrook, "Quantum generative adversarial learning," *Physical review letters*, vol. 121, no. 4, p. 040502, 2018.

[103] C. Zoufal, A. Lucchi, and S. Woerner, "Quantum generative adversarial networks for learning and loading random distributions," *npj Quantum Information*, vol. 5, no. 1, p. 103, 2019.

[104] M. S. Rudolph, N. B. Toussaint, A. Katabarwa, S. Johri, B. Peropadre, and A. Perdomo-Ortiz, "Generation of high-resolution handwritten digits with an ion-trap quantum computer," *Physical Review X*, vol. 12, no. 3, p. 031010, 2022.

[105] J. Li, R. O. Topaloglu, and S. Ghosh, "Quantum generative models for small molecule drug discovery," *IEEE transactions on quantum engineering*, vol. 2, pp. 1–8, 2021.

[106] K. Miyamoto, G. Morrás, T. S. Yamamoto, S. Kuroyanagi, and S. Nesseris, "Gravitational wave matched filtering by quantum monte carlo integration and quantum amplitude amplification," *Physical Review Research*, vol. 4, no. 3, p. 033150, 2022.

[107] S. Kumar and R. Tomar, "The role of artificial intelligence in space exploration," in *2018 International conference on communication, computing and internet of things (IC3IoT)*, pp. 499–503, IEEE, 2018.

[108] I. Khan, B. Heim, A. Neuzner, and C. Marquardt, "Satellite-based qkd," *Optics and Photonics News*, vol. 29, no. 2, pp. 26–33, 2018.

[109] S. Y. Chang, B. Le Saux, S. Vallecorsa, and M. Grossi, "Quantum convolutional circuits for earth observation image classification," in *IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium*, pp. 4907–4910, IEEE, 2022.

[110] A. Miroszewski, J. Nalepa, B. L. Saux, and J. Mielczarek, "Quantum machine learning for remote sensing: Exploring potential and challenges," *arXiv preprint arXiv:2311.07626*, 2023.

[111] LIGO, "Most precise ruler ever constructed." `https://www.ligo.caltech.edu/video/ligo20160211v6`, 2016. [Online; accessed January 2024].

[112] LIGO, "Black hole computer simulations help id third gravitational wave." `https://www.rit.edu/news/black-hole-computer-simulations-help-id-third-gravitational-wave`, 2017. [Online; accessed January 2024].

[113] A. Nitz, I. Harry, D. Brown, C. M. Biwer, J. Willis, T. D. Canton, C. Capano, T. Dent, L. Pekowsky, G. S. C. Davies, S. De, M. Cabero, S. Wu, A. R. Williamson, B. Machenschalk, D. Macleod, F. Pannarale, P. Kumar, S. Reyes, dfinstad, S. Kumar, M. Tápai, L. Singer, P. Kumar, veronica villa, maxtrevor, B. U. V. Gadre, S. Khan, S. Fairhurst, and A. Tolley, "gwastro/pycbc: v2.3.3 release of pycbc," Jan. 2024.

[114] G. Ashton, M. Hübner, P. D. Lasky, C. Talbot, K. Ackley, S. Biscoveanu, Q. Chu, A. Divakarla, P. J. Easter, B. Goncharov, *et al.*, "Bilby: A user-friendly bayesian inference library for gravitational-wave astronomy," *The Astrophysical Journal Supplement Series*, vol. 241, no. 2, p. 27, 2019.

[115] H. Qi and V. Raymond, "Python-based reduced order quadrature building code for fast gravitational wave inference," *Physical Review D*, vol. 104, no. 6, p. 063031, 2021.

[116] E. Thrane and C. Talbot, "An introduction to bayesian inference in gravitational-wave astronomy: parameter estimation, model selection, and hierarchical models," *Publications of the Astronomical Society of Australia*, vol. 36, p. e010, 2019.

[117] K. G. Wilson, "The renormalization group: Critical phenomena and the kondo problem," *Reviews of modern physics*, vol. 47, no. 4, p. 773, 1975.

[118] D.-X. Zhou, "Theory of deep convolutional neural networks: Downsampling," *Neural Networks*, vol. 124, pp. 319–327, 2020.

[119] M. S. ANIS *et Al*, "Qiskit: An open-source framework for quantum computing," 2021.

[120] J. M. Turney, A. C. Simmonett, R. M. Parrish, E. G. Hohenstein, F. A. Evangelista, J. T. Fermann, B. J. Mintz, L. A. Burns, J. J. Wilke, M. L. Abrams, *et al.*, "Psi4: an open-source ab initio electronic structure program," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 2, no. 4, pp. 556–565, 2012.

[121] R. G. Parr, "Density functional theory," *Annual Review of Physical Chemistry*, vol. 34, no. 1, pp. 631–656, 1983.

[122] M. Orio, D. A. Pantazis, and F. Neese, "Density functional theory," *Photosynthesis research*, vol. 102, pp. 443–453, 2009.

[123] A. Tranter, P. J. Love, F. Mintert, and P. V. Coveney, "A comparison of the bravyi–kitaev and jordan–wigner transformations for the quantum simulation of quantum chemistry," *Journal of chemical theory and computation*, vol. 14, no. 11, pp. 5617–5630, 2018.

[124] J. T. Seeley, M. J. Richard, and P. J. Love, "The bravyi-kitaev transformation for quantum computation of electronic structure," *The Journal of chemical physics*, vol. 137, no. 22, 2012.

[125] H. Basch, C. Hornback, and J. Moskowitz, "Gaussian-orbital basis sets for the first-row transition-metal atoms," *The Journal of Chemical Physics*, vol. 51, no. 4, pp. 1311–1318, 1969.

[126] R. Babbush, N. Wiebe, J. McClean, J. McClain, H. Neven, and G. K.-L. Chan, "Low-depth quantum simulation of materials," *Physical Review X*, vol. 8, no. 1, p. 011044, 2018.

[127] D. W. Berry, C. Gidney, M. Motta, J. R. McClean, and R. Babbush, "Qubitization of arbitrary basis quantum chemistry leveraging sparsity and low rank factorization," *Quantum*, vol. 3, p. 208, 2019.

[128] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, "Encoding electronic spectra in quantum circuits with linear t complexity," *Physical Review X*, vol. 8, no. 4, p. 041015, 2018.

[129] R. Babbush, D. W. Berry, I. D. Kivlichan, A. Y. Wei, P. J. Love, and A. Aspuru-Guzik, "Exponentially more precise quantum simulation of fermions in second quantization," *New Journal of Physics*, vol. 18, no. 3, p. 033032, 2016.

[130] D. P. DiVincenzo, "Fault-tolerant architectures for superconducting qubits," *Physica Scripta*, vol. 2009, no. T137, p. 014020, 2009.

[131] A.-S. Feiner and A. McEvoy, "The nernst equation," *Journal of chemical education*, vol. 71, no. 6, p. 493, 1994.

[132] F. Leng, Z. Wei, C. M. Tan, and R. Yazami, "Hierarchical degradation processes in lithium-ion batteries during ageing," *Electrochimica Acta*, vol. 256, pp. 52–62, 2017.

[133] R. Hausbrand, G. Cherkashinin, H. Ehrenberg, M. Gröting, K. Albe, C. Hess, and W. Jaegermann, "Fundamental degradation mechanisms of layered oxide li-ion battery cathode materials: Methodology, insights and novel approaches," *Materials Science and Engineering: B*, vol. 192, pp. 3–25, 2015.

[134] L. Wang, T. Maxisch, and G. Ceder, "A first-principles approach to studying the thermal stability of oxide cathode materials," *Chemistry of materials*, vol. 19, no. 3, pp. 543–552, 2007.

[135] A. Urban, D.-H. Seo, and G. Ceder, "Computational understanding of li-ion batteries," *npj Computational Materials*, vol. 2, no. 1, pp. 1–13, 2016.

[136] Y. Su, D. W. Berry, N. Wiebe, N. Rubin, and R. Babbush, "Fault-tolerant quantum simulations of chemistry in first quantization," *PRX Quantum*, vol. 2, no. 4, p. 040332, 2021.