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.

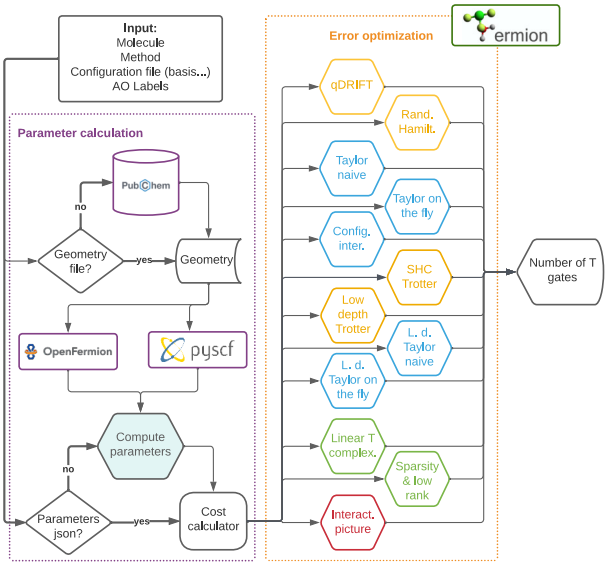

Figure 1: Flowchart of the architecture of our library, divided into two parts: the first one centered on the computation of the parameters needed for the cost estimate; and a second one on using such parameters to compute the number of T-gates. Methods are colored according to the Hamiltonian simulation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

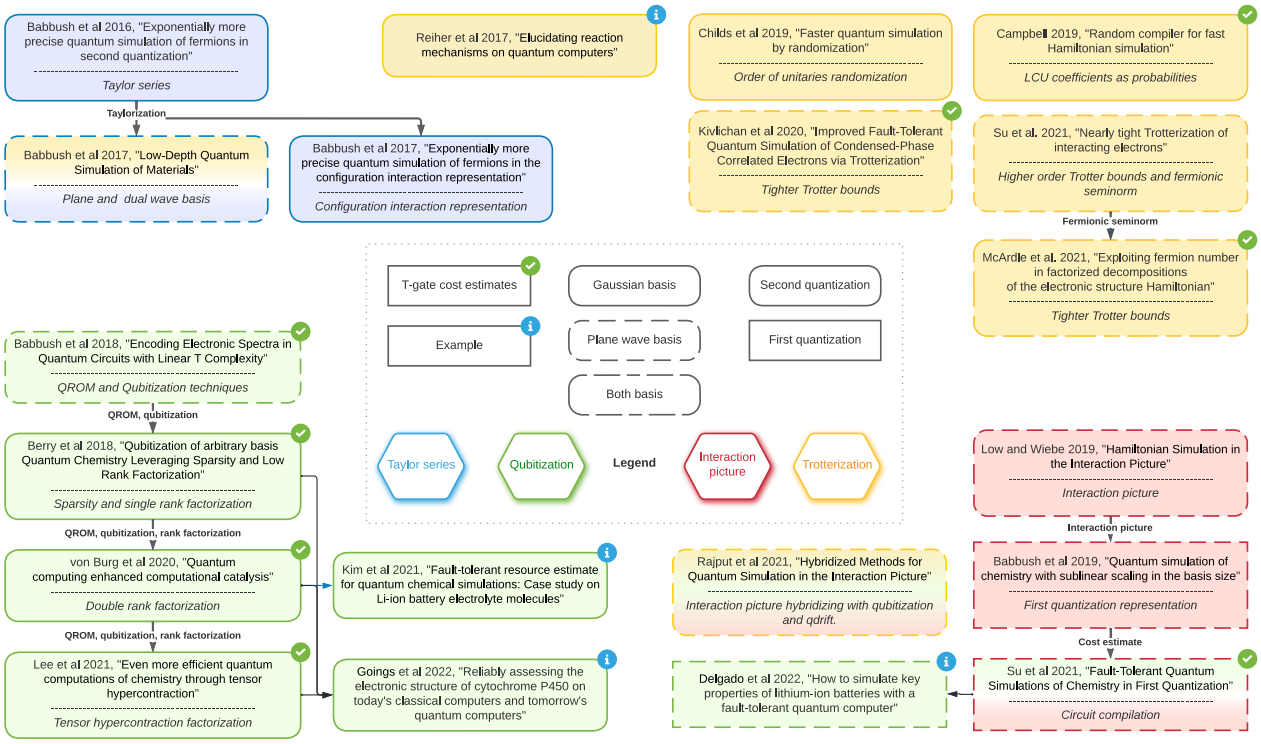

Figure 2: Diagram showing some of the main techniques involved in the development of post-Trotter Quantum Phase Estimation Techniques. Not shown in the picture but of great importance are the articles crystallizing the concept of 'Taylorization' [9] and 'qubitization', [44].

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]

$$\begin{aligned} e^{-i(A+B)\tau} \approx e^{-iA\tau} \lim_{\substack{K\to\infty \\ M\to\infty}} \sum_{k=0}^{K} \frac{(-i\tau)^k}{M^k k!} \sum_{m_1=0}^{M-1} \cdots \sum_{m_k=0}^{M-1} \\ \Big(e^{-i\tau(-1/2-m_k')A/M} B e^{-i\tau(m_k'-m_{k-1}')A/M} B \ldots \\ B e^{-i\tau(m_2'-m_1')A/M} B e^{-i\tau(m_1'+1/2)A/M}\Big) \\ = \left(\langle 0|\,\text{Prepare}_B^\dagger\right)^{\otimes K} \sum_{k=0}^{K} \frac{(-i\lambda_B\tau)^k}{M^k k!} \sum_{m_1,\ldots,m_k=0}^{M-1} \\ \Big(e^{-i\tau(M-1/2-m_k')A/M}\text{Select}_B e^{-i\tau(m_k'-m_{k-1}')A/M} \\ \text{Select}_B \ldots \text{Select}_B e^{-i\tau(m_2'-m_1')A/M}\text{Select}_B \\ e^{-i\tau(m_1'+1/2)A/M}\Big)\left(\text{Prepare}_B|0\rangle\right)^{\otimes K}, \end{aligned} \tag{18}$$

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

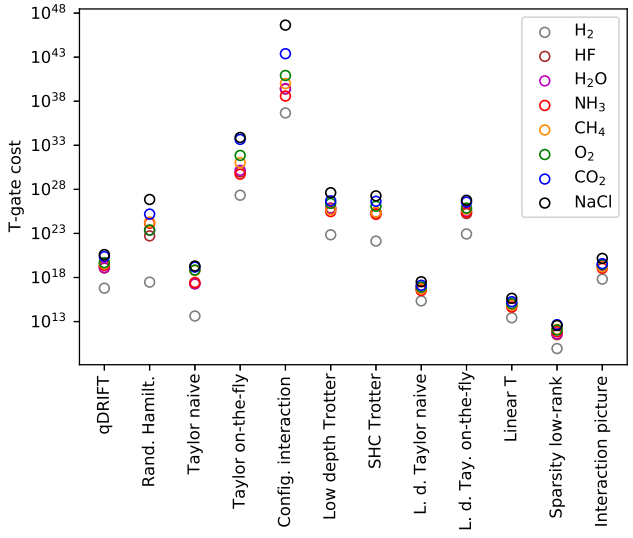

Figure 3: Representation of the results obtained for simple molecules with the results from table 6. We can see that choosing the right method greatly impacts the final cost of the Quantum Phase Estimation algorithm.

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').

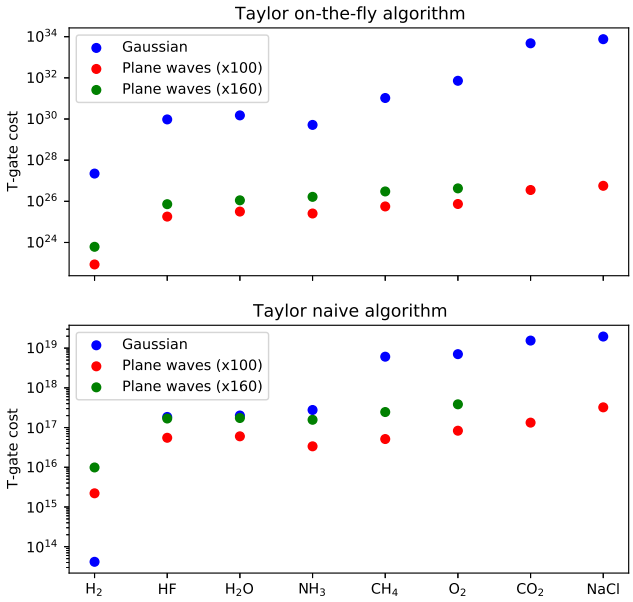

Figure 4: T-gate cost of performing the same algorithms making use of Taylorization as the main Hamiltonian simulation technique, over different molecules. The number of plane waves was chosen to be ≈ 100 or 160 times larger than Gaussian functions as recommended by Appendix E in [6]. The cost of computing 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:

$$e^{-iHt/r} \approx \tilde{U}_r := \sum_{k=0}^{K}\frac{(-iHt/r)^k}{k!} = \sum_{k=0}^{K}\sum_{\gamma_1,\ldots,\gamma_k=1}^{\Gamma}\frac{(-it/r)^k}{k!} w_{\gamma_1}\ldots w_{\gamma_k} H_{\gamma_1}\ldots H_{\gamma_k}, \quad (22)$$

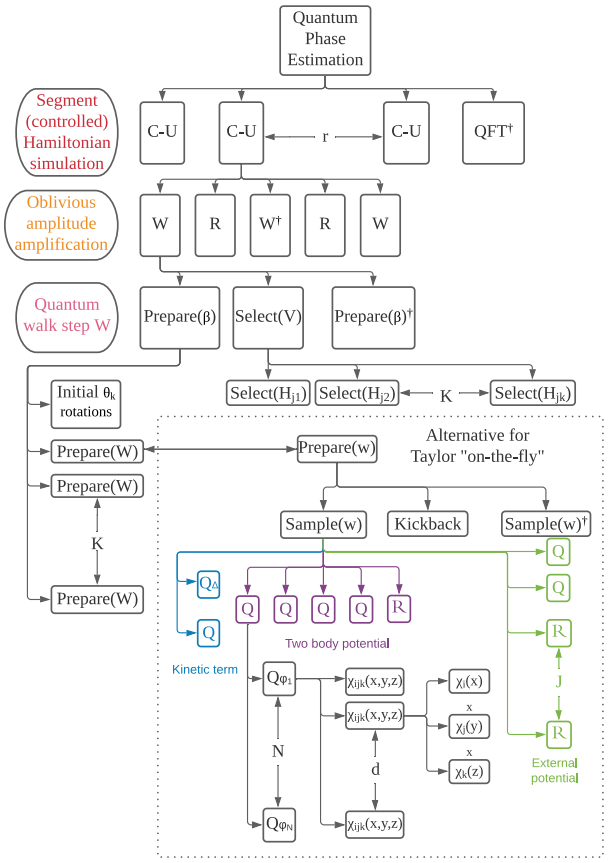

Figure 5: Abstraction level decomposition of the Taylor 'database' algorithm. The x-axis represents the time steps of the algorithm, while the y-axis is the abstraction level, higher meaning more abstract. In the lower box, we also depict the substitution one does to perform the alternative Taylor 'on-the-flight' 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

$$\text{sample}(w) \left|\gamma\right\rangle \left|\rho\right\rangle \left|0\right\rangle^{\otimes \lceil \log_2 M \rceil} = \left|\gamma\right\rangle \left|\rho\right\rangle \left|\tilde{w}_\gamma(\vec{z}_\rho)\right\rangle \tag{29}$$

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 $\left|l\right\rangle = \left|\gamma\right\rangle \left|m\right\rangle \left|\rho\right\rangle$, one performs *Kickback*:

$$\left|l\right\rangle \left|\tilde{w}_\gamma(\vec{z}_\rho)\right\rangle \to \begin{cases} \left|l\right\rangle \left|\tilde{w}_\gamma(\vec{z}_\rho)\right\rangle & \tilde{w}_\gamma(\vec{z}_\rho) > (2m - M)\zeta \\ i \left|l\right\rangle \left|\tilde{w}_\gamma(\vec{z}_\rho)\right\rangle & \tilde{w}_\gamma(\vec{z}_\rho) \leq (2m - M)\zeta \end{cases} \tag{30}$$

before uncomputing sample(w).

In summary, to prepare the amplitudes, one calculates sample(w) in the basis, performs (30) in a superposition of $\left|m\right\rangle$, and uncomputes the register prepared by sample(w). We will call such procedure Prepare(w):

$$\text{Prepare}(w) \left|0\right\rangle^{\otimes \lceil \log_2 L \rceil} = \sqrt{\frac{1}{\lambda'}} \sum_{l=1}^{L} \sqrt{\frac{\zeta \mathcal{V}}{\mu} w_{\gamma,m}(\vec{z}_\rho)} \left|l\right\rangle, \tag{31}$$

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 $\left|k\right\rangle$ 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.

$$\begin{aligned} Q &= \prod_{j=1}^{N} \left|j\right\rangle \left\langle j\right| \otimes Q_{\varphi_j}, \\ Q_{\varphi_j} \left|\rho\right\rangle \left|0\right\rangle^{\otimes \lceil \log_2 M \rceil} &= \left|\rho\right\rangle \left|\varphi_j(\vec{z}_\rho)\right\rangle. \end{aligned} \tag{34}$$

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.

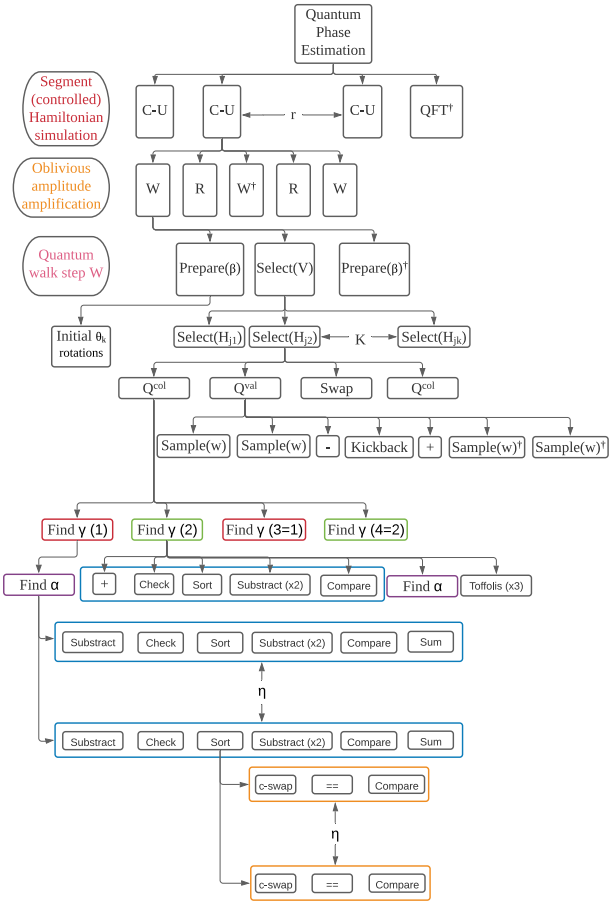

Figure 6: Abstraction level decomposition of the Configuration Interaction procedure [4]. The Sample operation shown is the same as in figure 5.

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 ×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

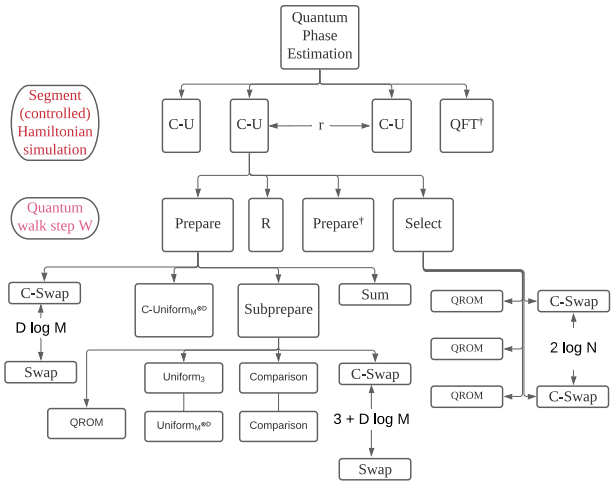

Figure 7: Abstraction level decomposition of the procedure [5].

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 ($D=3$)

$$M = (N/2)^{1/D}, \qquad f(p,\sigma) = \delta_{\sigma,\uparrow}M^D + \sum_{j=0}^{D-1}p_jM^j. \tag{59}$$

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

$$|0\rangle^{\otimes(2+\log_2 N)} \mapsto$$
$$\sum_{d=0}^{N-1} \Big(\tilde{U}(d) |\theta_d\rangle |1\rangle_U |0\rangle_T + \tilde{T}(d) |\theta_d^{(0)}\rangle |0\rangle_U |0\rangle_V + \tilde{V}(d) |\theta_d^{(1)}\rangle |0\rangle_U |1\rangle_V \Big) |d\rangle . \tag{63}$$

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

$$\varphi_{\boldsymbol{\nu}}(\boldsymbol{r}) = \sqrt{\frac{1}{\Omega}} e^{i\boldsymbol{k}_{\boldsymbol{\nu}} \cdot \boldsymbol{r}}, \quad \boldsymbol{k}_{\boldsymbol{\nu}} = \frac{2\pi \boldsymbol{\nu}}{\Omega^{1/3}}, \quad \boldsymbol{\nu} \in [-N^{1/3}, N^{1/3}]^3 \in \mathbb{Z}^3. \tag{64}$$

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,

$$H = \underbrace{\frac{1}{2N}\sum_{p,q,\nu,\sigma} k_\nu^2 \cos[k_\nu \cdot r_{q-p}] a_{p,\sigma}^\dagger a_{q,\sigma}}_{T}$$
$$\underbrace{-\frac{4\pi}{\Omega}\sum_{p,j,\sigma,\nu\neq 0}\left(\frac{\zeta_j \cos[k_\nu\cdot(R_j - r_p)]}{k_\nu^2}\right) n_{p,\sigma}}_{U} \quad (66)$$
$$\underbrace{+\frac{2\pi}{\Omega}\sum_{(p,\sigma)\neq(q,\sigma');\nu\neq 0}\frac{\cos[k_\nu\cdot r_{p-q}]}{k_\nu^2} n_{p,\sigma}n_{q,\sigma'}}_{V},$$

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

$$H = \frac{\pi}{2\Omega}\sum_{\substack{(p,\sigma)\neq(q,\sigma')\\ \nu\neq 0}}\frac{\cos[k_\nu\cdot r_{p-q}]}{k_\nu^2} Z_{p,\sigma}Z_{q,\sigma'}$$
$$\sum_{\substack{p,\sigma\\ \nu\neq 0}}\left(\frac{\pi}{\Omega k_\nu^2} - \frac{k_\nu^2}{4N} + \frac{2\pi}{\Omega}\sum_j \frac{\zeta_j\cos[k_\nu\cdot(R_j - r_p)]}{k_\nu^2}\right) Z_{p,\sigma}$$
$$+\frac{1}{4N}\sum_{\substack{p\neq q\\ \nu,\sigma}} k_\nu^2\cos[k_\nu\cdot r_{q-p}](X_{p,\sigma}Z_{p+1,\sigma}...Z_{q-1,\sigma}X_{q,\sigma}$$
$$+ Y_{p,\sigma}Z_{p+1,\sigma}...Z_{q-1,\sigma}Y_{q,\sigma}) + \sum_{\nu\neq 0}\left(\frac{k_\nu^2}{2} - \frac{\pi N}{\Omega k_\nu^2}\right) I. \quad (67)$$

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

$$H = \underbrace{FFFT^\dagger\left(\frac{1}{2}\sum_{\nu,\sigma} k_\nu^2 a_{\nu,\sigma}^\dagger a_{\nu,\sigma}\right)FFFT}_{T}$$
$$\underbrace{-\frac{4\pi}{\Omega}\sum_{p,j,\sigma,\nu\neq 0}\left(\frac{\zeta_j\cos[k_\nu\cdot(R_j - r_p)]}{k_\nu^2}\right) n_{p,\sigma}}_{U} \quad (68)$$
$$\underbrace{+\frac{2\pi}{\Omega}\sum_{(p,\sigma)\neq(q,\sigma');\nu\neq 0}\frac{\cos[k_\nu\cdot r_{p-q}]}{k_\nu^2} n_{p,\sigma}n_{q,\sigma'}}_{V}$$

where all the terms are diagonal. To implement this Hamiltonian, we need to perform a Fermionic Fast Fourier Transform (FFT) [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

$$e^{-iH\delta_t} \approx e^{-i(U+V)\delta_t/2}\cdot$$
$$FFFT^\dagger e^{-i(\delta_t/2)\sum_{\nu,\sigma} k_\nu^2 a_{\nu,\sigma}^\dagger a_{\nu,\sigma}} FFFT \quad (69)$$
$$\cdot\, e^{-i(U+V)\delta_t/2} + O(\delta_t^3),$$

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:

$$\text{Select}(H)\,|p,q,b\rangle\,|\psi\rangle = |p,q,b\rangle\otimes$$
$$\begin{cases} Z_p\,|\psi\rangle & p = q \\ Z_pZ_q\,|\psi\rangle & (b=0)\wedge(p\neq q) \\ X_p\vec{Z}X_q\,|\psi\rangle & (b=1)\wedge(p>q)\wedge(p\oplus q=0) \\ Y_p\vec{Z}Y_q\,|\psi\rangle & (b=1)\wedge(p<q)\wedge(p\oplus q=0) \\ |\psi\rangle & (b=1)\wedge(p\oplus q=1) \end{cases}. \quad (70)$$

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

$$W_{p,q,b} = \sum_{\nu\neq 0} W_{p,q,b,\nu}, \quad (71)$$

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

$$\begin{aligned} H &= \sum_{\sigma\in\{\uparrow,\downarrow\}} \sum_{p,q=1}^{N/2} h_{pq} a_p^\dagger a_q \\ &+ \frac{1}{2}\sum_{\alpha,\beta\in\{\uparrow,\downarrow\}} \sum_{p,q,r,s=1}^{N/2} h_{pqrs} a_{p,\alpha}^\dagger a_{q,\beta}^\dagger a_{r,\beta} a_{s,\alpha} \\ &= \sum_{\sigma\in\{\uparrow,\downarrow\}} \sum_{p,q=1}^{N/2} T_{pq} a_p^\dagger a_q \\ &+ \sum_{\alpha,\beta\in\{\uparrow,\downarrow\}} \sum_{p,q,r,s=1}^{N/2} V_{pqrs} a_{p,\alpha}^\dagger a_{q,\alpha} a_{r,\beta}^\dagger a_{s,\beta} \end{aligned} \tag{102}$$

The coefficients h_{pq} and h_{pqrs} are efficiently computable integrals. On the previous equation, the ordering $a^\dagger a^\dagger a a$ is called the 'physics notation' whereas the second ordering, $a^\dagger a a^\dagger a$ 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,

$$W g^{(l)} = w_l g^{(l)}; \quad W = \sum_{l=1}^{L} w_l g^{(l)} (g^{(l)})^T, \tag{103}$$

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

$$\frac{d}{dt}A(t) = \frac{i}{h}[H, A(t)] + \left(\frac{\partial A}{\partial t}\right)_H . \tag{118}$$

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 $\tilde{O}(N^2 t)$.

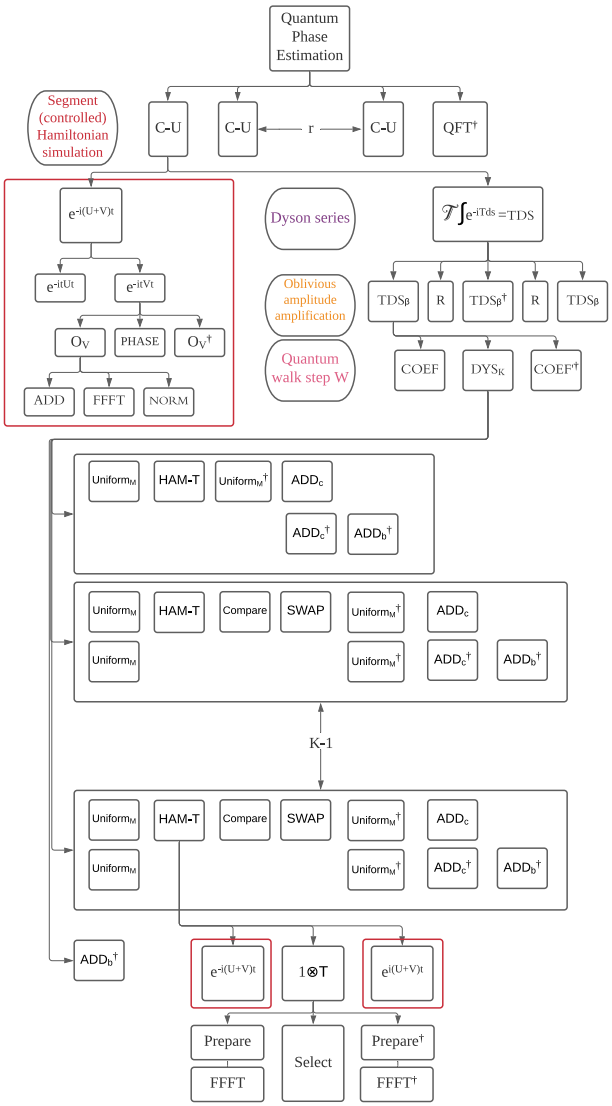

Figure 8: Abstraction level decomposition of the interaction picture protocol of [44]. The boxes in red represent the same protocol, only decomposed for one of them.

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 $||H(t)||$:

$$U(t) = \mathbf{1} - i\int_0^t H(t_1)dt_1 - \int_{t_2}\int_0^{t_2} H(t_2)H(t_1)dt_1dt_2 + i\int_{t_3}\int_{t_2}\int_0^{t_2} H(t_3)H(t_2)H(t_1)dt_1dt_2dt_3 + ... \tag{127}$$

We can rewrite the previous expression using the time ordering operator

$$U(t) = \mathcal{T}[e^{-i\int_0^t H(s)ds}] = \sum_{k=0}^{\infty}(-i)^k D_k. \qquad D_k = \frac{1}{k!}\int_0^t ...\int_0^t \mathcal{T}[H(t_k)...H(t_1)]d^k t. \tag{128}$$

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 = 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

$$\begin{aligned}V &= \sum_{(\vec{x},\sigma)\neq(\vec{y},\sigma')} V(\vec{x}-\vec{y})n_{\vec{x},\sigma}n_{\vec{y},\sigma'} \\ &= \sum_{\vec{k}}\tilde{V}(\vec{k})\tilde{\chi}_{\vec{k}}\tilde{\chi}^\dagger_{\vec{k}} + \sum_{\vec{p},\sigma}\left(\sum_{\vec{k}}\tilde{V}(\vec{k})\right)n_{\vec{p},\sigma}.\end{aligned} \tag{140}$$

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 $e^{-i(U+V)\tau}$.