Quantum simulation of chemistry via quantum fast multipole method

Dominic W. Berry,[1,*] Kianna Wan,[2,3] Andrew D. Baczewski,[4] Elliot C. Eklund,[5] Arkin Tikku,[5] and Ryan Babbush[2,†]

[1] *School of Mathematical and Physical Sciences, Macquarie University, New South Wales 2109, Australia*
[2] *Google Quantum AI, Venice, CA 90291, United States*
[3] *Stanford Institute for Theoretical Physics, Stanford University, Stanford, CA 94305, United States*
[4] *Quantum Algorithms and Applications Collaboratory, Sandia National Laboratories, Albuquerque, NM 87185, United States*
[5] *School of Chemistry, University of Sydney, NSW 2006, Australia*

(Dated: October 10, 2025)

Here we describe an approach for simulating quantum chemistry on quantum computers with significantly lower asymptotic complexity than prior work. The approach uses a real-space first-quantised representation of the molecular Hamiltonian which we propagate using high-order product formulae. Essential for this low complexity is the use of a technique similar to the fast multipole method for computing the Coulomb operator with $\widetilde{\mathcal{O}}(\eta)$ complexity for a simulation with η particles. We show how to modify this algorithm so that it can be implemented on a quantum computer. We ultimately demonstrate an approach with $t(\eta^{4/3}N^{1/3}+\eta^{1/3}N^{2/3})(\eta Nt/\epsilon)^{o(1)}$ gate complexity, where N is the number of grid points, ϵ is target precision, and t is the duration of time evolution. This is roughly a speedup by $\mathcal{O}(\eta)$ over most prior algorithms. We provide lower complexity than all prior work for $N < \eta^6$ (the regime of practical interest), with only first-quantised interaction-picture simulations providing better performance for $N > \eta^6$. As with the classical fast multipole method, large numbers $\eta \gtrsim 10^3$ would be needed to realise this advantage.

CONTENTS

* dominic.berry@mq.edu.au
† ryanbabbush@gmail.com

I. INTRODUCTION

Solving problems in chemistry is expected to be one of the most promising applications of quantum computing. Much of the early research in this direction focused on developing and improving quantum algorithms to compute energies of molecular systems using the quantum phase estimation algorithm. We often refer to the problem of computing molecular energies as the electronic structure problem. More recently, an effort has been made to rigorously explore the potential of quantum computing for applications in chemical dynamics where the focus is mainly computing correlation functions, reaction rates, and quantum yields. Both electronic structure and chemical dynamics rely on efficient quantum algorithms to simulate unitary evolution under the governing Hamiltonian. Enabling practical application for a wide range of problems in chemistry depends on developing and optimising algorithms for quantum simulation. Even more broadly, simulations of materials [1, 2] and degenerate plasmas [3, 4] make use of similar techniques, and often require studying systems with many more interacting electrons than is typical in chemistry. Thus, improvements to the asymptotic scaling of quantum simulation methods with the number of electrons might facilitate their application to not only larger molecular systems but also condensed phases.

The first quantum algorithms designed for quantum simulation were based on product formulae [5]. As the field of quantum algorithms matured, more advanced methods were introduced. These algorithms included techniques based on linear combinations of unitaries [6, 7], quantum walks [8], Taylor series [9], quantum signal processing [10], and block encoding [11]. A key breakthrough with these more recent methods is that they provide a relatively favourable complexity scaling in the simulation error ϵ going as $\mathcal{O}(\log(1/\epsilon))$. This is in contrast to product formulae, which scale as $\text{poly}(1/\epsilon)$. However, recent work shows that product formulae can be competitive with logarithmically scaling methods for simulation because they provide low error in practice [12, 13].

Simulating the types of Hamiltonians that are relevant in chemistry is made difficult by the fact that all pairwise interactions between charged particles need to be summed over in order to evaluate the Coulomb potential. For a system consisting of η electrons, this calculation yields an η^2 factor in the complexity of the algorithm. However, some approaches to simulation avoid the double sum, reducing this factor to $\mathcal{O}(\eta)$. For example, when the state of the molecule is encoded as a list of occupied orbitals in a first-quantised representation, it is possible to achieve a linear dependence in η by block encoding the Hamiltonian, which does not need the potential to be computed. Alternatively, a linear dependence in η can be achieved for the electronic structure problem in a second-quantised representation by calculating the Coulomb potential with a fast Fourier transform, avoiding the double sum. However, this comes with an additional cost that is linear the number of orbitals N due to second quantisation [14]. In cases where $N \gg \eta$, it is preferable to use first-quantised based approaches. We note that very recently, a first-quantised approach for simulating non-relativistic quantum electrodynamics was proposed that also avoids the $\mathcal{O}(\eta^2)$ overhead for the Coulomb potential [15].

Simulations based on product formulae in first quantisation [4, 16] show promise for better scaling than many of these algorithms, due to the Low *et al.* [17] bounds on the error in fermionic systems. A long-standing open question in quantum algorithms has been how to avoid the $\mathcal{O}(\eta^2)$ overhead for the Coulomb potential in this approach. In classical algorithms, the fast multipole method (FMM) [18–20] enables computation of the potential to be performed with complexity $\mathcal{O}(\eta\,\text{polylog}(1/\epsilon))$. The difficulty with applying the FMM in quantum algorithms is that a direct translation of the classical algorithm would result in data accesses in locations governed by the values in quantum registers. Each data access has complexity corresponding to the number of data locations, which here is η, increasing the overall complexity by this factor. Therefore, this approach results in an overall complexity that is larger than $\mathcal{O}(\eta^2)$, negating the speedup provided by FMM. The FMM approach discussed in Ref. [21], which considers the direct translation of the classical algorithm, incurs this extra overhead of η. In this instance, the overhead results implicitly from the use of QRAM. A detailed discussion of the issues associated with implementing an FMM quantum algorithm is provided by Babbush *et al.* in Ref. [3].

Our solution to recover the $\mathcal{O}(\eta \log \eta)$ scaling of the classical algorithm is related to the implementation of a quantum sort [22, 23] in which a sorting network is used to make comparisons at fixed locations so that the overhead from accessing quantum data is avoided. In the FMM algorithm, a tree structure is created, where each successive level of the tree partitions the simulation cell into increasingly smaller boxes, so that at the leaf level each box contains a constant number of particles. One difficulty involved with our sorting network approach is constructing a subroutine to retrieve information from the boxes so that the appropriate interactions can be evaluated. We solve this problem using multiple Morton orderings with shifts to ensure that all boxes in the interaction list are within a minimum distance in one order. The methods developed here to avoid the data-access overhead show promise for application to quantum versions of many other fast summation methods [24–30].

In the rest of this paper, we summarise the electronic structure Hamiltonian in Section II, and the FMM algorithm in Section III. In Section IV, we explain how to implement a quantum FMM algorithm for the case where the particles are assumed to be evenly distributed. We lift this assumption in Section V and explain how to implement an adaptive FMM algorithm so that the size of each box is adapted to the local density of particles. We estimate the total

complexity in Section VI, and conclude in Section VII.

II. SIMULATING QUANTUM CHEMISTRY IN REAL SPACE FIRST QUANTISATION

We begin by considering the simulation of the electronic structure problem defined on a spatial grid in first quantisation. Within the Born-Oppenheimer (BO) approximation, the electronic structure Hamiltonian is defined as

$$H = T + U + V + \sum_{l \neq \kappa = 1}^{\zeta} \frac{q_l q_\kappa}{2 \left\| R_l - R_\kappa \right\|} , \tag{1}$$

$$T = \sum_{i=1}^{\eta} \mathrm{QFT}_i \left(\sum_{\mathbf{p} \in G} \frac{\left\| \mathbf{k}_p \right\|^2}{2} |\mathbf{p}\rangle\langle\mathbf{p}|_i \right) \mathrm{QFT}_i^\dagger , \tag{2}$$

$$U = -\sum_{i=1}^{\eta} \sum_{l=1}^{\zeta} \sum_{\mathbf{p} \in G} \frac{q_l}{\left\| R_l - \mathbf{r}_p \right\|} |\mathbf{p}\rangle\langle\mathbf{p}|_i , \tag{3}$$

$$V = \sum_{i \neq j = 1}^{\eta} \sum_{\mathbf{p},\mathbf{q} \in G} \frac{1}{2 \left\| \mathbf{r}_p - \mathbf{r}_q \right\|} |\mathbf{p}\rangle\langle\mathbf{p}|_i |\mathbf{q}\rangle\langle\mathbf{q}|_j , \tag{4}$$

where QFT_i denotes the standard quantum Fourier transform applied to register i and $\| \cdot \|$ denotes the Euclidean norm. Further, l and κ index nuclear degrees of freedom, while i and j index electronic degrees of freedom. The positions of the nuclei and electrons in real space are denoted by R_l and $\mathbf{r}_p$, respectively. The atomic numbers of nuclei, which are used to express the nuclear charges, are denoted by q_l. Throughout this work, we use η and ζ to denote the numbers of electrons and nuclei, respectively. We work in atomic units such that $\hbar$, $4\pi\varepsilon_0$, and the mass and charge of the electron are unity.

The simulation cell is specified by its grid points and the associated frequencies in the dual space given by the QFT. In particular, we define

$$\mathbf{r}_p = \frac{\mathbf{p}\, \Omega^{1/3}}{N^{1/3}} , \qquad \mathbf{k}_p = \frac{2\pi\, \mathbf{p}}{\Omega^{1/3}} , \qquad \mathbf{p} \in G , \qquad G = \left[-\frac{N^{1/3}-1}{2}, \frac{N^{1/3}-1}{2} \right]^3 . \tag{5}$$

Here, Ω is the volume of the simulation cell and N is the number of grid points in the cell. The value of a grid point in real space is $\mathbf{r}_p$, and the associated value of the frequency is $\mathbf{k}_p$. For our implementation of the FMM, the position will be encoded by natural numbers in binary for each coordinate.

This is essentially a more precise reformulation of the same representation used by Kassal *et al.* [16] in the first work on quantum simulating chemistry in first quantisation. Our approach is then to perform simulation under the electronic structure Hamiltonian by using high-order product formulae and a split-operator Trotter step, that separately evolves under T, and then under $U + V$. The approach we use to evolve under T is essentially the same as that pursued by Kassal *et al.* [16], resulting in $\widetilde{\mathcal{O}}(\eta)$ complexity. In cases where ζ is small, it would be appropriate to perform simulation under the operator U using the approach from Kassal *et al.* [16] as well, which results in $\mathcal{O}(\eta\zeta)$ complexity. A significant innovation of our work is to compute V using a quantum version of the FMM, then perform time evolution with phase kickback. This approach can also be used to simulate U in cases with larger ζ, or for the internuclear potential in non-BO simulations.

Unlike the work of Kassal *et al.* [16], we propose to perform time evolution using high-order product formulae. The work of Low *et al.* [17] bounds the number of Trotter steps required for a split-operator approach in real-space second quantisation. This result for the number of Trotter steps also holds for real-space first quantisation (see the Supporting Information of Ref. [4], Section V.B.), and is

$$t \left(\frac{\eta^{2/3} N^{1/3}}{\Omega^{1/3}} + \frac{N^{2/3}}{\Omega^{2/3}} \right) \left(\frac{\eta N t}{\Omega \epsilon} \right)^{o(1)} = t \left(\eta^{1/3} N^{1/3} + \frac{N^{2/3}}{\eta^{2/3}} \right) \left(\frac{\eta N t}{\epsilon} \right)^{o(1)} \tag{6}$$

for time-evolution duration t and target precision ϵ. In the equality above we have made the assumption that $\Omega \propto \eta$. In this expression $o(1)$ indicates powers of the inverse of the order of the product formula, and so may be made arbitrarily small. Thus, the overall complexity will be given by Eq. (6) times the complexity of the Trotter step.

The complexity of the Trotter step is bottlenecked by the computation of V. The naïve method for this, described by Kassal *et al.* [16], has complexity $\widetilde{\mathcal{O}}(\eta^2)$. But we will instead show an approach based on the fast multipole method

Year	Reference	Primary innovation	Toffoli/T complexity
2017	Babbush *et al.* [34]	Using plane waves with Trotter	$\widetilde{\mathcal{O}}(\eta^2 N^{17/6}\sqrt{1+\eta\Omega^{1/3}/N^{1/3}}/(\Omega^{5/6}\epsilon^{3/2}))$
2017	Babbush *et al.* [34]	Using plane waves with LCU	$\widetilde{\mathcal{O}}((N^4/\Omega^{1/3}+N^{11/3}/\Omega^{2/3})/\epsilon)$
2018	Babbush *et al.* [35]	Linear scaling quantum walks	$\widetilde{\mathcal{O}}((N^{10/3}/\Omega^{1/3}+N^{8/3}/\Omega^{2/3})/\epsilon)$
2018	Low *et al.* [32]	Interaction picture with second quantisation	$\widetilde{\mathcal{O}}(N^{8/3}/(\Omega^{2/3}\epsilon))$
2018	Babbush *et al.* [33]	First quantised qubitisation	$\widetilde{\mathcal{O}}((\eta^3(N/\Omega)^{1/3}+\eta^2(N/\Omega)^{2/3})/\epsilon)$
2018	Babbush *et al.* [33]	Interaction picture with first quantisation	$\widetilde{\mathcal{O}}(\eta^3(N/\Omega)^{1/3}/\epsilon)$
2019	Kivlichan *et al.* [36]	Better Trotter steps	$\widetilde{\mathcal{O}}(N^3/(\Omega^{2/3}\epsilon^{3/2}))$
2019	Childs *et al.* [12]	Tighter Trotter bounds	$\mathcal{O}(N^{7/3+o(1)}/(\Omega^{1/3}\epsilon^{1+o(1)})$
2021	Su *et al.* [31]	Tighter Trotter bounds for plane waves	$N(\eta(N/\Omega)^{1/3}+(N/\Omega)^{2/3})N^{o(1)}/\epsilon^{1+o(1)}$
2023	Low *et al.* [17]	Tighter Trotter in real space	$N(\eta^{2/3}(N/\Omega)^{1/3}+(N/\Omega)^{2/3})(\eta N)^{o(1)}/\epsilon^{1+o(1)}$
2024	Rubin *et al.* [4]	Tighter Trotter bounds in first quantisation	$\eta^2(\eta^{2/3}(N/\Omega)^{1/3}+(N/\Omega)^{2/3})(\eta N)^{o(1)}/\epsilon^{1+o(1)}$
2025	Stetina & Wiebe [15]	Gauss' law as a constraint	$\eta^2(N/\Omega)^{2/3}(\eta N)^{o(1)}/\epsilon^{1+o(1)}$
2025	**This work**	quantum fast multipole	$\eta(\eta^{2/3}(N/\Omega)^{1/3}+(N/\Omega)^{2/3})(\eta N)^{o(1)}/\epsilon^{1+o(1)}$

TABLE I. Best quantum algorithms for phase estimating chemistry in a plane wave or real space basis. For phase estimation the time t is replaced with $1/\epsilon$ in the complexity. N is number of basis functions, $\eta < N$ is number of electrons, Ω is the computational cell volume, and ϵ is target precision.

with complexity $\widetilde{\mathcal{O}}(\eta)$. Multiplying the number of Trotter steps by $\widetilde{\mathcal{O}}(\eta)$, this will lead to an overall gate complexity

$$t\left(\frac{\eta^{5/3}N^{1/3}}{\Omega^{1/3}}+\frac{\eta N^{2/3}}{\Omega^{2/3}}\right)\left(\frac{\eta N t}{\Omega\epsilon}\right)^{o(1)}=t\left(\eta^{4/3}N^{1/3}+\eta^{1/3}N^{2/3}\right)\left(\frac{\eta N t}{\epsilon}\right)^{o(1)}. \tag{7}$$

The space complexity is $\mathcal{O}(\eta\log N\,\mathrm{polylog}(1/\epsilon))$ arising from the registers used to store the multipole information. The results developed here represent roughly a speedup by $\mathcal{O}(\eta)$ over the most comparable prior methods.

For a detailed comparison to prior work, see Table I. For high filling fractions where η and N are similar sized, this reduces to roughly the same complexity as the best prior second-quantised plane wave algorithms developed by Su *et al.* [31] and Low *et al.* [32]. In practice, it is expected that N is significantly larger than η. The algorithm with the best scaling for $N \gg \eta$ is the first-quantised interaction-picture method [33], but that has much larger scaling with η. As a result, the result presented here has better complexity than Ref. [33], and the lowest known complexity for any approach, whenever $N < \eta^6$.

III. FAST MULTIPOLE METHOD PRELIMINARIES

Here, we review the classical FMM for a system consisting of η point particles. For the sake of clarity, we describe the non-adaptive case that is suitable for systems where the particles are roughly uniformly distributed within a cubic simulation cell. Let q_i and $\mathbf{r}_i$ ($i \in [\eta] \coloneqq \{1,\ldots,\eta\}$) be the charge and position of the ith particle, respectively. Our objective is to compute the total potential energy given by

$$V=\frac{1}{2}\sum_{i\in[\eta]}\sum_{\substack{j\in[\eta]\\ j\neq i}}q_i\Phi(\mathbf{r}_i,\mathbf{r}_j)q_j=\sum_{i\in[\eta]}q_iV_i\,, \tag{8}$$

for some kernel function Φ, where $V_i=\frac{1}{2}\sum_{j\in[\eta]\setminus\{i\}}\Phi(\mathbf{r}_i,\mathbf{r}_j)q_j$. In the case where all particles are electrons and we are interested in the Coulomb potential in 3D, we have $q_i=-1$ for all i and $\Phi(\mathbf{r}_i,\mathbf{r}_j)=1/\|\mathbf{r}_i-\mathbf{r}_j\|$ (using atomic units). Computing V exactly involves summing over all pairs of particles, which requires $\mathcal{O}(\eta^2)$ operations.

Instead, the FMM separates the potential experienced by each particle V_i into near-field and far-field contributions,

$$V=\sum_{i\in[\eta]}q_iV_i\ =\sum_{i\in[\eta]}q_i(V_{i,\mathrm{near}}+V_{i,\mathrm{far}}(\epsilon)). \tag{9}$$

Here, $V_{i,\mathrm{near}}$ is evaluated exactly via summation over a number of geometrically close particles. The number of close particles is bounded above by a constant that is at most $\mathcal{O}(\log\eta)$. In contrast, $V_{i,\mathrm{far}}$ is evaluated approximately via a Taylor expansion that rapidly converges to within a target error ϵ thanks to the low off-diagonal rank of $\Phi(\mathbf{r}_i,\mathbf{r}_j)$. The potential V is evaluated in $\mathcal{O}(\eta\log\eta\,\mathrm{polylog}(1/\epsilon))$ operations using an appropriate choice of metric to distinguish near- and far-field contributions. This metric is defined in terms of a hierarchical tree-like division of the simulation cell, and the efficient evaluation of V is typically framed in terms of the upward and downward traversal of this tree.

B_ℓ	set of boxes on level ℓ
$NN(b)$	nearest neighbours of box b
$P(b)$	parent box of b
$C(b)$	set of child boxes of b
$I(b)$	interaction list for b
$p(b)$	indices of particles contained in b
$\mathbf{c}_b$	position of the centre of box b

TABLE II. Summary of notation for boxes.

A. Defining the tree

An L-level hierarchical decomposition of the simulation cell defines a metric that determines whether the interaction between any pair of particles is accounted for in $V_{i,\text{near}}$ or $V_{i,\text{far}}$. An example decomposition is illustrated in Fig. 1. This decomposition naturally maps onto a tree that is organised so that the root node (1st level) encompasses the entire simulation cell, the leaf nodes (Lth level) partition the cell into n_b boxes, and nodes at intermediate levels partition the cell into boxes with double the edge length, up to the root. In d dimensions, there are $2^{d(\ell-1)}$ boxes at the ℓth level of this tree. While we will sometimes use a $d = 2$ quadtree for illustrative purposes in figures, we are generally concerned with a $d = 3$ octree in the evaluation of V for electronic structure problems.

The governing principle in FMM is that the interaction between particles in boxes that are well separated can be efficiently evaluated by

(1) aggregating the distribution of "source" particles (w.l.o.g., those indexed in the sum over j in Eq. (8)) into multipole expansions at all levels of the tree,

(2) translating the multipole expansions defining the attendant potential contribution into local (Taylor) expansions about boxes containing "observer" particles (w.l.o.g., those indexed in the sum over i in Eq. (8)), and

(3) disaggregating the local expansions into the potential experienced by any observer particle (i.e., $V_{i,\text{far}}$).

The remaining contributions to the potential experienced by any observer particle (i.e., $V_{i,\text{near}}$) are due to particles in boxes that are *not* well separated. These interactions are evaluated exactly.

While the near-field interactions are strictly accounted for among leaf-level boxes, evaluating the far-field interactions takes advantage of the fact that boxes separated by increasingly larger distances are more efficiently accounted for at higher levels of the tree. Performing these higher-level calculations requires defining the interaction list for any given box b – the set of boxes that are not too close to b, but also not too far away. In particular, two boxes are in each other's interaction list when they are not nearest neighbours but their parent boxes (at the next level up) are. This is illustrated in Figure 1.

To make this concept more rigorous, we define a few concepts. At a given level ℓ, let B_ℓ be the set of all boxes at this level. The size of B_ℓ in 3D is $|B_\ell| = 8^{\ell-1}$. For a box $b \in B_\ell$, we denote the set of nearest neighbours to b as $NN(b)$. Here, we define the nearest neighbours of b to be all boxes that share an edge or vertex with b. For $\ell \neq 1$, the size of $NN(b)$ in 3D is $|NN(b)| \leq 26$. The parent box $P(b) \in B_{\ell-1}$ is the box one level up from b such that b is fully enclosed by the volume of $P(b)$. We also define the child boxes of b to be boxes at level $\ell + 1$ that are enclosed by the volume of b. The set of child boxes of b is denoted $C(b)$. Note that $b \in C(P(b))$ is always true. Additionally, let $p(b)$ be a set of integers that index the particles within a given box b. The interaction list of b is given by

$$I(b) = C(NN(P(b))) - NN(b) - b, \tag{10}$$

where the final term is needed so that b itself is not included in the interaction list. Finally, let $\mathbf{c}_b$ be a vector representing the position of the centre of box b. The notation introduced here is summarised in Table II.

The interaction list of any box has at most $6^3 - 3^3 = \mathcal{O}(1)$ boxes in 3D. To see this, note that along any dimension the width of $C(NN(P(b)))$ is 6 boxes wide. In 3D, this yields 6^3 total boxes. Then, notice that the width of $NN(b)$, including the centre box b, along any dimension is 3, for a total of 3^3 in 3D. Finally, the total number of boxes in the interaction list is given by the difference, $6^3 - 3^3 = 189$. There can be fewer boxes near the boundaries of the region.

B. Traversing the tree

Here, we consider an illustrative zeroth-order (monopole only) implementation, and provide additional details required to achieve arbitrary ϵ using higher orders of the multipole expansion in Appendix A. In Algorithm 1, we

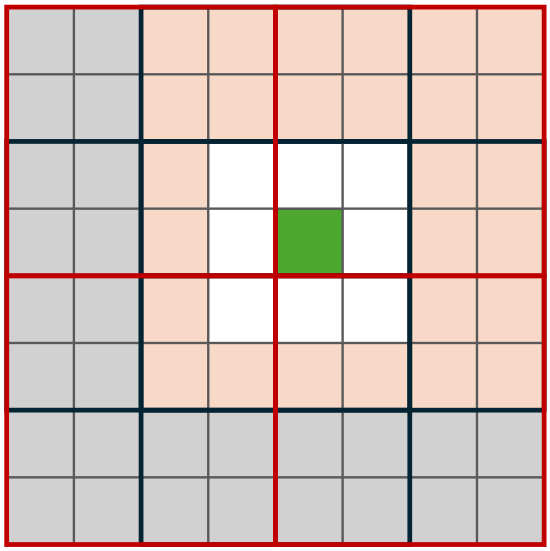

FIG. 1. Illustration of hierarchical division of a simulation cell, as well as the interaction list and neighbours for an exemplary (green) box in the $\ell = 4$th level of a tree. Here, the level-1 box is the complete region. The four level-2 boxes are outlined by the red lines. The 16 level-3 boxes are shown by the thick black lines and the level-2 partition. The 64 level-4 boxes are shown by the thin black lines and the level-3 partition. The box of interest b is represented by the green square. The surrounding white squares are neighbours of b, and the boxes in b's interaction list are shown in orange. The grey squares are neither nearest neighbours, nor in the interaction list. A two-dimensional simulation cell is presented for simplicity of presentation.

provide pseudocode that implements the classical FMM algorithm and returns an approximation to V within given error ϵ. The algorithm consists of two passes through the hierarchical tree in which every box at levels $\ell \geq 3$ is iterated through. In the first pass, the objective is to compute the charge Q_b for each box so that it can be used in the second pass to compute an approximation of V_i. The first pass starts at level $\ell = L$ and works up to $\ell = 3$. At $\ell = L$ the total charge Q_b of the particles in each box b is computed. Then, at $\ell = L - 1$, the total charge of each box is computed by adding the charges of the box's 8 child boxes computed previously. We continue in this way until $\ell = 3$ is reached. If there are at most $c = \mathcal{O}(1)$ particles per box at level L, computing Q_b for a given box requires $\mathcal{O}(1)$ operations. The total number of operations is then $\mathcal{O}(n_b)$ (omitting the ϵ-dependence).

The objective of the second pass is to compute an approximation to V using the monopoles computed in the first pass. This is done starting at level 3 and working up to level L. The first two levels are skipped because none of the boxes in B_1 and B_2 are sufficiently separated from the remaining boxes for the multipole approximation to be valid. This is the reason the box charges are only calculated up to level 3 in the first pass. For all other levels, the interactions between a given box b and the boxes in its interaction list are computed using the multipole approximation. At the leaf-level, the interactions between particles occupying the same box are computed exactly using the Coulomb potential. Additionally, the interactions between particles in a given leaf-level box and particles in the nearest-neighbour boxes are computed exactly. All of the interactions computed throughout the second pass are aggregated in a variable V that is returned after the algorithm completes.

Because each interaction list contains $\mathcal{O}(1)$ boxes, and each level-L box has $\mathcal{O}(1)$ neighbours (each with $\mathcal{O}(1)$ particles), the second pass also requires $\mathcal{O}(n_b)$ operations in total. If the particles are roughly uniformly distributed, then L can be chosen such that $n_b = \mathcal{O}(\eta)$, which enables the calculation of V with complexity $\mathcal{O}(\eta)$. The complexity has further polynomial factors of $\log(1/\epsilon)$ for the full multipole calculation. For more general distributions, an adaptive subdivision of boxes can be used to limit the number of boxes that need be considered, providing similar complexity.

IV. QUANTUM FMM FOR EVENLY DISTRIBUTED PARTICLES

The difficulty in constructing a quantum implementation of the FMM is that we need to place electrons into boxes according to their positions, which are stored in quantum registers. This means that we need to access data according to a value stored in a quantum register. For each electron, we need to run through $\mathcal{O}(\eta)$ boxes to determine the box that a given electron belongs to. Because there are η electrons this approach results in $\mathcal{O}(\eta^2)$ cost, which we are trying to avoid. We first illustrate the principles used to avoid $\mathcal{O}(\eta^2)$ scaling for the simpler case where the particles are assumed to be evenly distributed. In Section V, we modify our approach to allow for adaptive grids which lets us treat

Algorithm 1: Classical FMM (monopole only)

First pass: compute charges
1. **for** $\ell \leftarrow L$ **to** 3
  for $b \in B_\ell$
    initialise $Q_b \leftarrow 0$
    if $\ell \neq L$
      $Q_b \leftarrow \sum_{a \in C(b)} Q_a$.
    else
      $Q_b \leftarrow \sum_{j \in p(b)} q_j$
    end
  end
end

Second pass: evaluate potential
initialise $V \leftarrow 0$
Compute far-field contribution using box-box potentials
2. **for** $\ell \leftarrow 3$ **to** L
  for $b \in B_\ell$
    initialize $V_b \leftarrow 0$
    for $a \in I(b), a > b$
      $V_b \leftarrow V_b + Q_b \Phi(\mathbf{c}_a, \mathbf{c}_b) Q_a$
    end
    $V \leftarrow V + V_b$
  end
end

Compute near-field contribution using particle-particle potentials
3. **for** $b \in B_L$
  initialise $V_b \leftarrow 0$
  for $i \in p(b)$
    for $j \in p(b)$, $j > i$
      Compute Coulomb interactions between particles within b
      $V_b \leftarrow V_b + q_i \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j$
    end
    for $n \in NN(b), n > b$
      for $j \in p(n)$
        Compute Coulomb interactions between particles within b and n
        $V_b \leftarrow V_b + q_i \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j$
      end
    end
  end
  $V \leftarrow V + V_b$
end
Return V

the general case where the evenly distributed assumption is lifted. We note that within the BO approximation the positions of the nuclei are given by classical registers and there is no difficulty moving data for the nuclei. For non-BO simulations the registers storing the positions of the nuclei would need to be moved into boxes as well. However, this can be achieved with a similar approach as for the electron registers.

A. Quantum registers for FMM data storage

Our implementation of the quantum FMM algorithm uses several quantum registers to store information used throughout the algorithm. For a given number of electrons η, let $n_b = \mathcal{O}(\eta)$ so that the maximum number of electrons per box at the leaf level is c and the overall complexity is $\widetilde{\mathcal{O}}(\eta)$. For a given box b, we have the register $|Q_b\rangle$ which stores multipole information up to the chosen order of truncation. For the purpose of the following discussion, $|Q_b\rangle$ contains the total charge (monopole) of b.

In addition to $|Q_b\rangle$, boxes at the leaf level have the additional registers:

- $|\mathbf{r}_i\rangle,\ i \in p(b)$ – c registers storing the position of each particle

- $|q_i\rangle,\ i \in p(b) - c$ qubits which flag whether each of the $|\mathbf{r}_i\rangle$ has been populated with information for an electron.

That is, at the leaf level each of the n_b boxes is given c registers, which are used to store the positions of the electrons that occupy the box. Additionally, each of these $n_b c$ position registers has an associated flag register that is set to $|1\rangle$ if the position register is occupied, and is $|0\rangle$ otherwise. More generally, we can allow $|q_i\rangle$ to encode the charge for a nucleus, if we include nuclei in the FMM. We group together these position and flag registers as what we will call a "particle" register, and will consider operations moving data that move both the position and flag registers together.

Although we have described the particle registers with respect to the leaf-level boxes, they are used for boxes at all levels of the tree. In particular, the registers of box b at the leaf level are also used for the boxes in $P(b)$. Hence, each box in $\ell = L - 1$, has $2^d c$ particle registers for dimension d. For the description of the quantum FMM it is convenient to regard these registers to be arranged in the same way as the spatial locations.

Each position $\mathbf{r}$ is in a space G of N grid points. We encode each $\mathbf{r} \in G$ as $|\mathbf{r}\rangle = |x\rangle\,|y\rangle\,|z\rangle$ where each $|\alpha\rangle$ ($\alpha \in \{x, y, z\}$) is a binary representation of the spatial coordinate α. This uses $3\lceil \log(N^{1/3}) \rceil$ qubits (where we adopt the usual quantum information convention that logs are to base 2). Position registers that do not encode an electron are set to the all-zero state. When interleaving the bits for the Morton ordering, we encode the position as $|\mathbf{r}\rangle = \bigotimes_{k=1}^{n_p} |x_k\rangle\,|y_k\rangle\,|z_k\rangle$ for $k \in \{1, 2, ..., L, ..., n_p\}$, where $n_p = \lceil \log(N^{1/3}) \rceil$.

We store the position by natural numbers starting from zero, rather than numbers in the range $[-(N^{1/3} - 1)/2, (N^{1/3} - 1)/2]$ as indicated by G in Eq. (5). This is because the natural numbers are more convenient for defining the tree structure for the FMM. This encoding is for the position, but for the kinetic part of the Hamiltonian we need momenta that are symmetric about zero. In the implementation of the Hamiltonian this can be accounted for in how the Fourier transform is implemented.

With this encoding of the position, we can easily determine which box an electron should be placed in based on the encoding of its position. For example, at the highest level the triple of the first (most significant) bits (x_1, y_1, z_1) of the x, y and z coordinates of the electrons naturally partitions the box into 8 equal sections. Each of these 8 boxes is further subdivided into 8 more boxes by the triple of next most significant bits (x_2, y_2, z_2). This can then be continued up to the triple (x_L, y_L, z_L). As a result, we do not need additional resources to determine which box a given electron should be moved to.

B. Moving electron data to the correct registers

Before computing the potential using the FMM, we need to move the data for each of the η electrons into the position registers for the appropriate leaf-level boxes. This is achieved in a recursive way, where if the electron data is in the correct boxes at level ℓ, then within each box the electron data is moved into the correct child boxes by a procedure of sorting the data and swapping it into the appropriate child box. We first describe how to move electron position data into the correct boxes for the 1D case, then generalise to 3D. Figure 2 illustrates this procedure for the 1D case.

First, the system registers storing the electron positions are used as the first η of the particle registers, with the flag registers initialised to $|1\rangle$. The remaining $n_b c - \eta$ particle registers are initialised to the all-zero state, so $|\mathbf{0}\rangle = |0 \ldots 0\rangle$ and $|0\rangle$ for the position and flag registers, respectively. Next, the particle registers are sorted according to the position. The complexity of the sort is $\mathcal{O}(\eta \log \eta)$ comparisons and controlled swaps, which yields a gate complexity of $\mathcal{O}(\eta \log \eta \log N)$. The convention for each sort is that the flag registers are used to ensure that the populated registers are moved to the left. Note that we are sorting the particle registers, so the flag registers are moved with the position registers.

After sorting, we divide the complete simulation cell (level-1 box) into two equally sized level-2 boxes. Then, the data in the registers is swapped so that the electron positions are placed in the appropriate level-2 boxes. To achieve this task, we run through all $n_b c/2$ registers in the left box, or $\min(n_b c, \eta)$ locations accounting for the fact that not all of these registers need contain data for electrons. For each position register, we check the most significant qubit of $\mathbf{r}$. If the qubit is in the state $|0\rangle$, the electron belongs to the left box and does not need to be swapped. If the qubit is in the state $|1\rangle$, the electron belongs to the right box, and we swap it. If the electron was in register m, we swap the data to register $m + n_b c/2$, which is register m of the right box. As before, we are swapping the particle register, with both the position and flag registers. The complexity of the controlled swaps is $\mathcal{O}(n_b c \log N) = \mathcal{O}(\eta \log N)$.

When the data of a register is swapped to the right, it is important that the register it is swapped with does not contain an electron position (so the flag register is in the $|0\rangle$ state). This can be proven for our procedure in the following way. If there are initially no electrons in the registers for the right box, then it is trivially true. If there are electrons initially in the registers for the right box, then all data in registers m to $n_b c/2 - 1$ needs to be swapped to the right (counting from 0). This is because the registers are initially sorted. If it were the case that there was an electron already stored in register $m + n_b c/2$, then it would mean that there were more than $n_b c/2$ electrons that

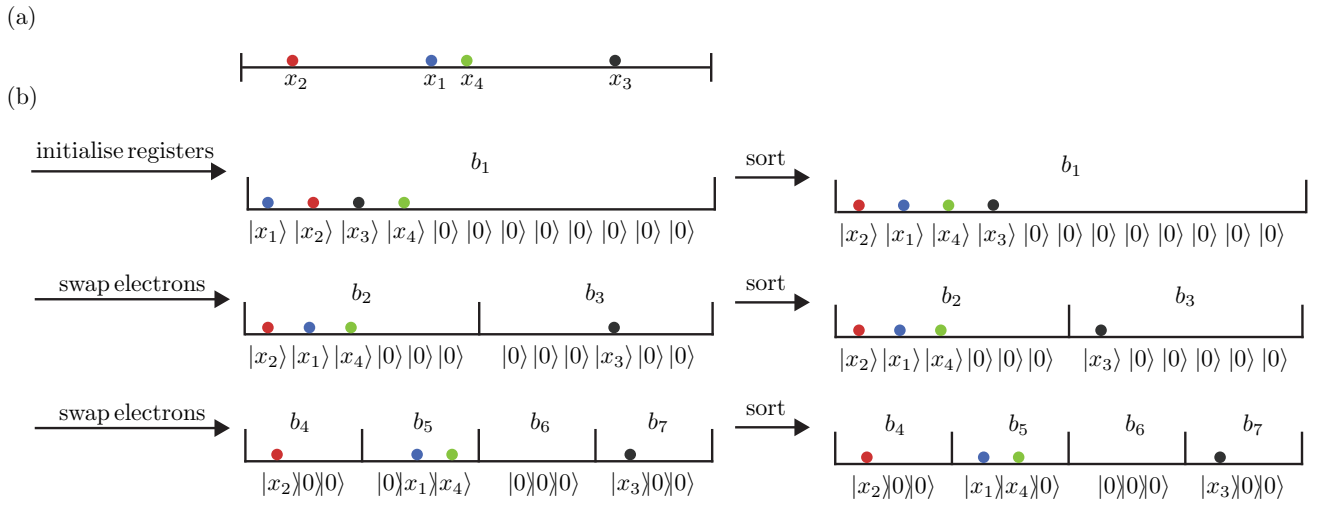

FIG. 2. 1D demonstration of Algorithm 2. Here $\eta = 4$, $L = 3$, $c = 3$, and $n_b = 4$. (a) The positions of the electrons are shown on the real space grid, where, x_i is the position of the ith electron. (b) Snapshots of the boxes and position registers throughout Algorithm 2. The label of a given box is shown above and the corresponding registers are shown below. The flag registers are not depicted. Each row iterates through the outer-most loop of Algorithm 2 for $\ell = 1, 2, 3$. For a given ℓ, the iteration starts by swapping position data into the correct registers. For $\ell = 1$, this step is replaced by an initialisation step. Next, the positions are sorted within each box. This procedure is repeated until $\ell = L$ is reached.

should be in the right box. This is more electrons than can be stored in the registers, violating the even distribution assumption in this section. Hence, this procedure never swaps data for an electron from the right box into the left box.

After swapping the data to the correct registers, we perform a sort on the registers of the right box with complexity $\mathcal{O}(\eta \log \eta \log N)$. The left box is already guaranteed to be sorted because the order was not changed. Each of the two level-2 boxes is then subdivided into two level-3 boxes. We then perform the controlled swaps within each level-2 box to ensure the electrons are within the correct level-3 boxes. We proceed in this way until we swap the data into the correct level-L boxes. There are $\log n_b$ levels, each with leading-order complexity for the sort $\mathcal{O}(\eta \log \eta \log N)$, as well as $\mathcal{O}(\eta \log N)$ complexity for controlled swaps, for a total complexity

$$\mathcal{O}(\eta \log^2 \eta \log N)\,. \tag{11}$$

In three (or any number of) dimensions, we can apply the result for one dimension by choosing an ordering for the boxes. For example, we can use a Morton ordering where the bits of the x, y, and z directions are interleaved. Alternatively, we can give the bits for x first, then y, then z. For either ordering, the same procedure can be used as in the 1D case to move the electrons to the correct boxes. The only difference is that the interpretation of the subdivision of the region into two corresponding to child boxes no longer holds.

The complete procedure is described in Algorithm 2. In this algorithm, it is assumed that there are initially $8^L c$ position and flag registers. The η electron position registers are input as part of these $8^L c$ registers with the associated flag registers set to $|1\rangle$. All other position registers are set to $|\mathbf{0}\rangle$, and flag registers to $|0\rangle$. All sorts used within this algorithm are quantum sorts, and move the flag registers together with the position registers, as well as moving occupied registers to the left.

Here, the quantity ℓ counts the number of times the region has been divided by 2, rather than the level, and b is an integer corresponding to which region is being considered, rather than a box of the octree. The index j counts through the registers in the sorted sequence, and p_j is the value in that register interpreted as an integer, which may, for example, correspond to interleaved bits of x, y, and z for Morton ordering. The test $p_j \geq (j_0 + w/2)/c \times N/2^{3(L-1)}$ is used to check if the register is in the wrong half of the region.

C. Calculation of the potential given correct data locations

Given that the electrons are placed in the correct registers, it is straightforward to calculate the potential via the FMM, directly translating the steps in Algorithm 1 into quantum gates. The algorithm as described above can be

Algorithm 2: Sorting electron registers

```
for ℓ ← 0 to 3(L − 1) − 1
    for b ← 0 to 2^ℓ − 1
        w ← c × 2^{3(L−1)−ℓ}
        j_0 ← bw
        Sort registers j_0 to j_0 + w − 1.
        for j ← j_0 to j_0 + w − 1
            if p_j ≥ (j_0 + w/2)/c × N/2^{3(L−1)}
                Swap register j with register j + w/2.
            end
        end
    end
end
```

directly translated into a quantum algorithm using standard methods of performing coherent quantum arithmetic. The parts of the calculation that directly use the information for the electrons in the boxes are the first step of loop 1 (which sums the charges in a box for $\ell = L$), and loop 3 (which sums the potential over electrons in neighbouring boxes).

For these calculations we iterate over all c registers in each box, but only add the contribution if the corresponding flag qubit stores $|1\rangle$. In particular, for $Q_b \leftarrow \sum_{j \in p(b)} q_j$, we sum the values stored in these flag registers which gives the total number of electrons in the box. Then, for $V_b \leftarrow V_b + q_i \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j$, we perform a Toffoli on the flag registers for electrons i and j, and use that to control addition of the potential term $\Phi(\mathbf{r}_i, \mathbf{r}_j)$.

The electron-nuclear potential for U is easily accounted for by adding the nuclear charges for the boxes in loop 1 of Algorithm 1, and for $V_b \leftarrow V_b + q_j \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j$ in loop 3, summing over nuclear charges as well. The sum over nuclear charges is easily performed because the positions are given classically, so it is known which nuclei are within each box without requiring any quantum data accesses.

The remaining steps involve computations on other registers associated with boxes, which are independent of how many particles are in each box. These include summing charges from child boxes (loop 1), and summing contributions to the potential from boxes in the interaction list (loop 2). These operations all involve additions, except loop 2 also requires a multiplication. The value of $\Phi(\mathbf{c}_a, \mathbf{c}_b)$ used in loop 2 does not need to be computed using coherent arithmetic, because we classically iterate over the box numbers a and b, and the values of $\Phi(\mathbf{c}_a, \mathbf{c}_b)$ can be precalculated classically. The only case where potentials need be calculated via coherent arithmetic is for the interparticle potentials in loop 3.

Overall the complexity of these calculations is as follows.

- The complexity of adding the numbers of electrons in the n_b boxes is $\mathcal{O}(n_b c)$.
- For adding the charges, at level ℓ, there are $3(L-\ell) + \lceil \log c \rceil$ qubits to perform arithmetic on, and $n_b / 2^{3(L-\ell-1)}$ additions to perform. Summing this complexity over $\ell = 1$ to $L-1$ gives total complexity $\mathcal{O}(n_b \log c)$.
- At level ℓ, Q_a has $3(L-\ell) + \lceil \log c \rceil$ qubits, so the complexity of multiplying by the potential with $\mathcal{O}(\log(1/\epsilon))$ qubits is $\mathcal{O}([3(L-\ell) + \log c] \log(1/\epsilon))$. Summing this complexity over the boxes and levels gives total complexity $\mathcal{O}(n_b \log c \log(1/\epsilon))$.
- For each leaf-level box there are $\mathcal{O}(c^2)$ pairwise potentials to approximate, for a total of $\mathcal{O}(n_b c^2)$ for all boxes. The values of $\Phi(\mathbf{r}_i, \mathbf{r}_j)$ can be approximated by using a QROM for interpolation combined with Newton's method [37]. In practical usage the number of iterations of Newton's method can be constant, but the number of iterations for error ϵ is $\mathcal{O}(\log\log(1/\epsilon))$. Each step of Newton's method uses multiplication, with complexity of $\mathcal{O}(\log^2(1/\epsilon))$, yielding a total complexity

$$\mathcal{O}\left(n_b c^2 \log^2(1/\epsilon)\, \log\log(1/\epsilon)\right). \tag{12}$$

Out of these complexities, the dominant complexity is that from summing the pairwise potentials between electrons, which for $n_b = \mathcal{O}(\eta)$ and c a constant is

$$\mathcal{O}(\eta \log^2(1/\epsilon)\, \log\log(1/\epsilon)). \tag{13}$$

Note that in this analysis we allow the error from each step in the calculation to be $\mathcal{O}(\epsilon)$, which is why the number of bits for the arithmetic is taken to be $\mathcal{O}(\log(1/\epsilon))$. In practice the error in each step will need to be smaller to ensure

the total error is no larger than ϵ, but that has no effect on the complexity given in the form of an order scaling in $\log(1/\epsilon)$. When including higher orders in the multipole expansion, the order should be taken to be $\mathcal{O}(\log(1/\epsilon))$, the number of multipole moments scales as the square of the order, and each should be given to $\mathcal{O}(\log(1/\epsilon))$ bits. This results in a higher power in the scaling with $\log(1/\epsilon)$; see Appendix A for details.

V. ADAPTIVE BOXES FOR ARBITRARY DISTRIBUTIONS OF PARTICLES

In the case where particles are distributed in an uneven way, the boxes are chosen in an adaptive way. That is, instead of subdividing the boxes down to a fixed level, subdivide boxes that contain multiple particles. That enables boxes at a very small level, but no more than η of these boxes need be retained because the others have no particles. The difficulty is that the boxes need to be determined based on the numbers in registers, which would mean data accesses at locations according to these values in a simple translation of the classical method. That would result in an overall complexity at least η^2, eliminating the speedup. In order to implement the procedure with a speedup we again need a way to perform the computation with data accesses at fixed locations.

A. Calculating charge / multipole information

First we consider how to calculate the charge and multipole information for the boxes, then consider the more difficult question of how to compute the potential information, which requires locating data for boxes in the interaction list. The general principle we use is, for each particle include a set of data registers for every box that the particle is contained in. In contrast to the non-adaptive case, we do not include a flag qubit or additional unoccupied particle registers. We first explain the method in one dimension, then how to generalise it to multiple dimensions. We also describe the method for the electron-electron potential V first, then the amendment for the electron-nuclear potential.

The difficulty is primarily in finding the data locations for boxes in order to compute the results for multiple boxes. The method we use to solve this is to copy data along the chain of registers for electrons, so that data for one box is copied to a neighbouring box in order to perform the calculation. In one dimension, the electron registers are first sorted into order. Then we can have a sequence of electron registers that are part of one box at level ℓ, followed by electron registers in a neighbouring box (or possibly another box).

In one dimension, we would calculate the charge for the box at level $\ell-1$ by adding the charges of the child boxes at level ℓ. Each box at level ℓ has a register containing that charge information for level ℓ, but that information needs to be copied over to the registers for charges in the next child box in that level in order to compute the charge at level $\ell-1$. Similarly, the multipole information needs to be copied over to compute the multipole information at level $\ell-1$.

In particular, let us denote the charge information for the box that electron register j is in at level ℓ by $Q(j,\ell)$. We explain the procedure for charge, and the procedure for multipole information is equivalent, except the operations to combine the multipole information are more complicated. Let us also denote by $f_b(j,\ell)$ the bits up to ℓ of the electron register, which is the number of the box at level ℓ. We take L to be the total number of bits for the positions of the electron registers, so all electrons are in different boxes at level L. The procedure is then as in Algorithm 3. This is the quantum implementation of the first pass of the FMM in Algorithm 1.

This algorithm first sets the charge at the leaf level. Then in 2(a) it runs forward through the sequence of registers, and checks if the neighbouring registers are for the same box at level ℓ but different boxes at level $\ell+1$. If they are, then it copies the charge from the left level-$\ell+1$ box into the register for the right level-$\ell+1$ box, but in the register being used for level-ℓ information. The other possibility is if the least significant bits of $f_b(j+1,\ell+1)$ and $f_b(j,\ell+1)$ are equal to 1 (checked using mod 2), in which case we have a pair of registers in the right level-$\ell+1$ box. Then the appropriate information will be in $Q(j,\ell)$, and is just copied into $Q(j+1,\ell)$.

In this way, every time we hit a boundary between two level-$\ell+1$ boxes that need their charges summed, we copy the charge information from the left box into the sequence of registers in the right box. If it happened that there were no electrons in the left box, then the $f_b(j+1,\ell)=f_b(j,\ell)$ test would fail, and we would just leave the $Q(j+1,\ell)$ register as zero. Then 2(b) just does the same procedure running from right to left, copying information from the right boxes into the left boxes. This ensures that at level ℓ, $Q(j,\ell)$ contains the charge from the *other* level-$\ell+1$ box. Then 2(c) adds the charges, so $Q(j,\ell)$ contains the total charge for the level-ℓ box (adding the two charges of the child boxes at level $\ell+1$). For the multipole information the procedure is identical, except the operation in 2(c) to combine multipole information is not just addition.

The same principle can be used in higher dimensions. In one dimension the division into boxes is simply repeated division of boxes into two. In 2D we can take the complete region, divide it into two in the x-direction, then into two in the y-direction, then in the x direction and so on. The principle is that we are still dividing into two at each stage,

Algorithm 3: Calculating charge information

```
1. for j ← 0 to η − 1
   | Q(j, L) ← 1        (corresponding to a single charge for this finest-level box)
end
2. for ℓ ← L − 1 to 3 by −1
   | (a) for j ← 0 to η − 2
   |    | if f_b(j + 1, ℓ) = f_b(j, ℓ)
   |    |    | if f_b(j + 1, ℓ + 1) ≠ f_b(j, ℓ + 1)
   |    |    |    Q(j + 1, ℓ) ← Q(j, ℓ + 1)
   |    |    | else if f_b(j, ℓ + 1) mod 2 = 1
   |    |    |    Q(j + 1, ℓ) ← Q(j, ℓ)
   |    |    | end
   |    | end
   | end
   | (b) for j ← η − 2 to 0 by −1
   |    | if f_b(j, ℓ) = f_b(j + 1, ℓ)
   |    |    | if f_b(j, ℓ + 1) ≠ f_b(j + 1, ℓ + 1)
   |    |    |    Q(j, ℓ) ← Q(j + 1, ℓ + 1)
   |    |    | else if f_b(j, ℓ + 1) mod 2 = 0
   |    |    |    Q(j, ℓ) ← Q(j + 1, ℓ)
   |    |    | end
   |    | end
   | end
   | (c) for j ← 0 to η − 1
   |    Q(j, ℓ) ← Q(j, ℓ) + Q(j, ℓ + 1)
   | end
end
```

even though we would be aiming to divide into 4 and use the information from four squares at each level to compute Q for one level up. Similarly, in 3D we would subdivide the region in the x direction, then y direction then z, and repeat.

This is equivalent to interleaving the bits of the x, y, and z components of the electron position. Interleaving the bits of the three coordinates and sorting yields the particles in Morton ordering. When we do this we can perform exactly the same procedure as in the 1D case with the interleaved bits of x, y, z replacing the bits of x. At each stage we would add charge from two neighbouring regions rather than 8, but three levels give addition of the charges from 8 sub-cubes. Algorithm 3 is unchanged, except $f_b(j, \ell)$ is now given by the bits up to ℓ of the electron register with bits for the x, y, and z coordinates interleaved, and L would be the total number of bits (but three times the number of levels).

Note that when we use the registers with $Q(j, \ell)$ for calculating the potential, we want ℓ to be the usual FMM level. This means that we need to adjust the indexing, which just corresponds to a choice of labelling of the registers, rather than requiring quantum operations. We assume that this has been done when using $Q(j, \ell)$ in later steps.

B. Calculating the potential

The principle is to run through the registers for all electrons and copy the information for one region over to the neighbouring region. This is similar to what we have already done for the charge, but when calculating the potential we have the further complication that it is more difficult to find the region. When considering the charge, we were able to combine two regions in the x direction, then y direction, then z direction. For the potential we need to find particles in regions in the interaction list.

The geometry of the interaction list is illustrated in Fig. 1 for two dimensions. The box of interest (in green) at level-ℓ is part of a level-$\ell - 1$ box indicated with thick black lines. The other level-ℓ boxes inside that box are neighbours, so we do not add the potential. That is in contrast to the method for charge, where we add charges for sub-boxes. The boxes where we do add the potentials, indicated in orange, are part of the neighbouring level-$\ell - 1$ boxes.

Not only that, but we have a row of three of these larger boxes. This means that they are not just contained in one level-$\ell - 2$ box, but overflow into neighbouring boxes at that level. Nevertheless, we can still use a similar approach for copying information from neighbouring regions, which we will initially explain for the 1D case.

1. Accessing interaction list for 1D case

For each electron, we have extra registers for one, two, and three positions over; we will call these $Q_1(j,\ell)$, $Q_2(j,\ell)$, and $Q_3(j,\ell)$. We run through the electrons in sequence, and for each we check if the following electron corresponds to a new region; that is $f_b(j,\ell) \neq f_b(j+1,\ell)$. The cases are then as follows.

1. If $f_b(j,\ell) = f_b(j+1,\ell)$, then both electrons are in the same region, and we just directly copy over the charge information. That is, $Q_k(j+1,\ell) = Q_k(j,\ell)$ for $k \in \{1,2,3\}$.

2. If $f_b(j+1,\ell) = f_b(j,\ell)+1$, then the next electron is in the next region, and we copy the charge information to the next electron's register for the neighbouring region; that is, $Q_1(j+1,\ell) = Q(j,\ell)$. We also copy the charge information for one location over to that for two locations over, and that for two regions over to that for three regions over, so $Q_2(j+1,\ell) = Q_1(j,\ell)$ and $Q_3(j+1,\ell) = Q_2(j,\ell)$.

3. If $f_b(j+1,n) = f_b(j,n)+2$, then the next electron is two regions displaced, and we directly copy into the register for two regions over, so $Q_2(j+1,\ell) = Q(j,\ell)$. We also copy the information for one region over to that for three regions over, so $Q_3(j+1,\ell) = Q_1(j,\ell)$. We leave $Q_1(j+1,\ell)$ zeroed, because that region is missing.

4. If $f_b(j+1,n) = f_b(j,n)+3$, then the next electron is three regions displaced, and we directly copy into the register for three regions over, so $Q_3(j+1,\ell) = Q(j,\ell)$. We leave both $Q_1(j+1,\ell)$ and $Q_2(j+1,\ell)$ zeroed, because those regions are missing.

5. If $f_b(j+1,\ell) > f_b(j,\ell)+3$, then the next electron is more than three regions displaced, and $Q_1(j+1,\ell)$, $Q_2(j+1,\ell)$, and $Q_3(j+1,\ell)$ can be left zeroed.

After doing this, we have associated with each electron, registers giving the charge information for the boxes to the left of it, which can be used in computing the potential. We can then invert the above procedure to erase those working ancillas. To describe the procedure in a more general case where we copy information from up to K boxes over, we use registers $Q_k(j,\ell)$ for $k = 1$ to K, and it is also convenient to use $Q_0(j,\ell)$ to denote the charge information for the box itself. Then the general procedure for copying information from neighbours is as shown in Algorithm 4.

Algorithm 4: Copying information for K-neighbours

```
for j ← 0 to η − 2
    if f_b(j+1,ℓ) = f_b(j,ℓ)
        for k ← 1 to K
            Q_k(j+1,ℓ) ← Q_k(j,ℓ)
        end
    else if f_b(j+1,ℓ) ≤ f_b(j,ℓ) + K
        Δk ← f_b(j+1,ℓ) − f_b(j,ℓ)
        for k ← Δk to K
            Q_k(j+1,ℓ) ← Q_{k−Δk}(j,ℓ)
        end
    end
end
```

In this algorithm, we run through the registers from 0 to $\eta-2$, with the requirement that the registers are sorted so $f_b(j,\ell)$ (bits up to ℓ of the position) are in ascending order. Then $f_b(j+1,\ell) = f_b(j,\ell)$ implies that the next register is in the same box, so we copy over the information for the neighbours of that box. If $f_b(j+1,\ell) \leq f_b(j,\ell)+K$, then we have stepped over to another box, but by no more than K. Then we copy over the appropriate neighbour information, skipping some if $\Delta k > 1$, which means that there were empty boxes skipped over in going from particle j to $j+1$.

To access information from all boxes in the interaction list, we can also perform the copying in the reverse direction. Although copying in the reverse direction is needed to access information from all boxes in the interaction list, it is not necessary to calculate the potential. This is because, for each box a in the interaction list of box b, b will be in the interaction list of a. Copying the information in the reverse direction will only give box-box potentials that have already been accounted for, so is not needed.

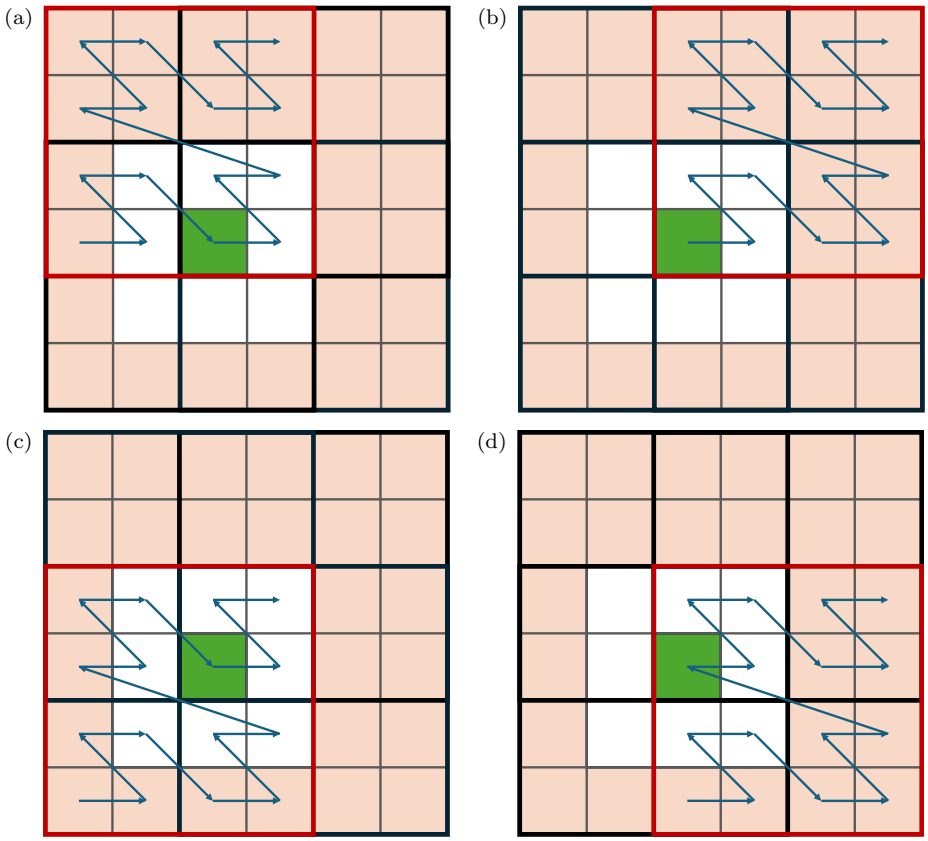

FIG. 3. The interaction list and neighbours used for FMM. The green square is the box b of interest, the white squares are neighbours of b, and the boxes in the interaction list are shown in orange. The heavy black lines indicate the boxes for the next level up. The red lines indicate the box two levels up containing the box of interest, with parts (a) to (d) indicating the four alternatives. The blue zig-zag lines indicate the Morton ordering of boxes within that higher-level box.

2. *Accessing interaction list for higher dimensions*

In order to obtain the information from the boxes in the interaction list in two or three dimensions, we can use a similar approach interleaving bits as we use for charge, but it is more challenging because the boxes in the interaction list need to be accessed by taking a 1D path through this higher-dimensional space. To address this issue, we use Morton ordering, as well as bit shifts. In particular, the Morton ordering is obtained by interleaving the bits for the x, y, and z directions and sorting.

In Figure 3 the level-$\ell-2$ box is shown in red, with four different alternatives for where it can be situated relative to the box of interest, b. The Morton ordering means that iterating through the particles in the sorted list will run through the boxes in the sequence indicated by the blue arrows. Therefore, we can use the approach above for 1D, but use temporary registers to account for boxes in the interaction list displaced by up to 15 positions (in 2D) or 63 positions (in 3D).

Note that the boxes in the interaction list may be before or after the target box b in the ordering, so we would need to go through the sorted list both forwards and backwards to access information from all boxes in the interaction list. However, there is a similar consideration here as for the 1D case. Because $a \in I(b) \implies b \in I(a)$, going through the sorted list backwards would only yield box-box potentials that have already been accounted for when running through the list forwards, and so the reverse direction is not needed.

Using just a single Morton ordering does not guarantee that all the boxes in the interaction list are within a fixed distance from b in that ordering. This is an issue that has been considered in the classical computing literature, and can be solved by using multiple orderings. For example, Refs. [38, 39] use multiple Hilbert orderings, and Ref. [40] uses multiple Morton orderings. Here, we use a slightly different approach to those, which is more easily implemented in a quantum algorithm.

In particular, the Morton ordering is more easily implemented than the Hilbert ordering, because it can be achieved by simply interleaving the bits of x, y, and z, then applying the quantum sort. To achieve the multiple orderings, we apply combinations of increments in the x, y, and z directions, and sort again. For a level-$\ell-2$ box, bit number $\ell-3$

is identical for all positions within the box. That is, bit number $\ell-3$ is identical for all x-coordinates, bit number $\ell-3$ is identical for all y-coordinates, and so on. For example, if $\ell-2=2$ (the first subdivision of the complete region), then bit number $\ell-3=1$ is the first (most significant) bit.

Now the effective level-$\ell-2$ box can be changed by incrementing the coordinates. If the level-$\ell-2$ box is that shown in Figure 3(a), then adding 1 to bit number $\ell-2$ for the x-coordinate will shift the effective level-$\ell-2$ box over to that shown in Figure 3(b). For example, if $\ell-2=2$ and the first 3 bits of the x-coordinates are

$$000,\ 001,\ 010,\ 011,\ 100,\ 101, \tag{14}$$

then you can see that the first bit is the same for the first four (corresponding to the first four level-ℓ boxes in the diagram). Adding 1 to bit number $\ell-2=2$ gives

$$010,\ 011,\ 100,\ 101,\ 110,\ 111, \tag{15}$$

so the four boxes on the right have the same first bit, and are within the same effective level-$\ell-2$ box.

In this way, by the 4 combinations of incrementing x and y (or x, y, and z in the 3D case), the effective level-$\ell-2$ box can be shifted to each of the possible locations. This will be true regardless of where the initial (unshifted) position of the level-$\ell-2$ box is. For example, if the position is that illustrated in Figure 3(b), then the increment in the x-coordinate will shift the effective box to that shown in Figure 3(a).

For each shift, the bits of x, y, and z are interleaved and the sort is performed to give the Morton ordering with a shift. Performing this with each combination of shifts guarantees that each box in the interaction list is within a fixed distance of b in at least one ordering. In particular, the result can be given as follows, with the notation $[a,b]$ indicating the set corresponding to the range of integers.

Lemma 1 (Morton orderings with shifts)**.** *For $\mathbf{p}\in[0,2^n-1]^d$, let*

$$D_p := \left\{\mathbf{q}\in[0,2^n-1]^d \;\middle|\; ||\lfloor p_j/2\rfloor - \lfloor q_j/2\rfloor|\le 1,\ \forall j\in[1,d]\right\}, \tag{16}$$

and $M:\mathbb{N}^d\to\mathbb{N}$ be the function applying the Morton ordering to a vector of integers. Then for all $\mathbf{q}\in D_p$, there exists a vector of increments $\mathbf{z}\in\{0,2\}^d$ such that

$$|M(\mathbf{p}+\mathbf{z} \bmod 2^n) - M(\mathbf{q}+\mathbf{z} \bmod 2^n)| \le 4^d-1\,. \tag{17}$$

Proof. First, note that points satisfying $\lfloor \mathbf{p}/2^m\rfloor = \lfloor \mathbf{q}/2^m\rfloor$ will satisfy $\lfloor M(\mathbf{p})/2^{dm}\rfloor = \lfloor M(\mathbf{q})/2^{dm}\rfloor$. This is because the set of points satisfying $\lfloor \mathbf{p}/2^m\rfloor = \lfloor \mathbf{q}/2^m\rfloor$ are those within a cube of side length 2^m corresponding to the first $n-m$ bits of each coordinate of $\mathbf{p}$ being equal to those of $\mathbf{q}$. The Morton ordering groups all 2^{dm} points with these same initial bits together, because it corresponds to sorting with the bits interleaved. Because there are 2^{dm} points, they can be separated by no more than $2^{dm}-1$ in the Morton ordering.

Now, given a point $\mathbf{q}\in D_p$, take

$$\mathbf{z} = 2|\lfloor \mathbf{p}/4\rfloor - \lfloor \mathbf{q}/4\rfloor|\,. \tag{18}$$

Since $||\lfloor p_j/2\rfloor - \lfloor q_j/2\rfloor|\le 1$, it must also be true that $||\lfloor p_j/4\rfloor - \lfloor q_j/4\rfloor|\le 1$, and so this vector is within the required set $\{0,2\}^d$. For any given j, if $z_j=0$, then $\lfloor p_j/4\rfloor = \lfloor q_j/4\rfloor$. Then it is clear that

$$\lfloor (p_j+z_j \bmod 2^n)/4\rfloor = \lfloor p_j/4\rfloor = \lfloor q_j/4\rfloor = \lfloor (q_j+z_j \bmod 2^n)/4\rfloor\,. \tag{19}$$

On the other hand, if $z_j=2$, then put $p_j=4a+r$ and $q_j=4b+s$ with $r,s\in\{0,1,2,3\}$. For $p_j<q_j$, then $b=a+1$ and $r\in\{2,3\}$, $s\in\{0,1\}$. This restriction on r,s is required so that $||\lfloor p_j/2\rfloor - \lfloor q_j/2\rfloor|\le 1$ for the requirement that $\mathbf{q}\in D_p$. Then

$$p_j+z_j = 4a+r+2 = 4(a+1)+r-2 = 4b+r-2\,, \tag{20}$$
$$q_j+z_j = 4b+s+2\,. \tag{21}$$

The conditions $r\in\{2,3\}$, $s\in\{0,1\}$ imply that $r-2\in\{0,1\}$, $s+2\in\{2,3\}$. Note that neither p_j+z_j or q_j+z_j can increase above 2^n-1, and so the modular addition does not affect the outcome, and

$$\begin{aligned}\lfloor (p_j+z_j \bmod 2^n)/4\rfloor &= \lfloor (p_j+z_j)/4\rfloor\\ &= \lfloor (4b+r-2)/4\rfloor\\ &= b\end{aligned}$$

$$
\begin{aligned}
&= \lfloor (4b + s + 2)/4 \rfloor \\
&= \lfloor (q_j + z_j)/4 \rfloor \\
&= \lfloor (q_j + z_j \bmod 2^n)/4 \rfloor . \qquad (22)
\end{aligned}
$$

The reasoning for $z_j = 2$ and $p_j > q_j$ is identical with the roles of p_j and q_j reversed. Thus in all cases

$$\lfloor (p_j + z_j \bmod 2^n)/4 \rfloor = \lfloor (q_j + z_j \bmod 2^n)/4 \rfloor . \qquad (23)$$

In turn, using the result given at the start of the proof, this implies that $\lfloor M(\mathbf{p}+\mathbf{z})/4^d \rfloor = \lfloor M(\mathbf{q}+\mathbf{z})/4^d \rfloor$, and the points $M(\mathbf{p}+\mathbf{z} \bmod 2^n)$ and $M(\mathbf{q}+\mathbf{z} \bmod 2^n)$ cannot be separated by any more than $4^d - 1$, which is the result to be proven for the Lemma. □

The interpretation of this Lemma is that $\mathbf{p}$ is the vector for a box number at level ℓ for dimension d, and D_p is the set of boxes that are neighbours of $\mathbf{p}$ or in the interaction list (as well as $\mathbf{p}$ itself). This means that it is within a level-$\ell - 1$ box (the reason for the division by 2 and floor) displaced by no more than one position in any direction. Less significant bits for positions within that box number are not included in this Lemma. Then $\mathbf{z}$ is the vector of displacements by 2, corresponding to shifting by the size of a level-$\ell - 1$ box. The additions are modulo the size of the entire region of 2^n, then Eq. (17) bounds the difference between $\mathbf{p}$ and $\mathbf{q}$ in the shifted Morton ordering.

This choice of shifted Morton orderings then enables us to copy to each box the data from all boxes in its interaction list for the calculation of the potential. The complete procedure is given in Algorithm 5, which is the quantum implementation of loop 2 of the FMM algorithm in Algorithm 1. At each level, we apply the increment and the quantum sort according to the Morton ordering. With that sort, we go through the list of particles, copying the information from neighbouring boxes at level ℓ. As discussed above, we need only go through the list in one direction, because the reverse direction would only yield box-box potentials already accounted for.

Algorithm 5: Calculation of potential in dimension d

```
K ← 4^d − 1
V ← 0
for ℓ ← 3 to L
    for z ∈ {0,2}^d
        Add z to box number at level ℓ and sort according to Morton ordering.
        Apply Algorithm 4 to copy charge information.
        for j ← 0 to η − 1
            if j is first particle in level-ℓ box
                for k ← 1 to K
                    if Q_k(j,ℓ) contains charge information for a box in the interaction list not already added
                        V ← V + Q(j,ℓ)Φ(c_a, c_b)Q_k(j,ℓ)
                    end
                end
            end
        end
        Invert Algorithm 4 to erase Q_k(j,ℓ) registers. Invert Morton ordering sort and subtract z.
    end
end
```

At each step, we check whether the potential was added at a previous step, because the set of Morton orderings results in the charge information for each box in the interaction list being included multiple times. In Algorithm 5, the quantity $\mathbf{c}_b$ is the position of the centre of the level-ℓ box that particle j is in, and $\mathbf{c}_a$ is the centre of the box that the information in $Q_k(j,\ell)$ corresponds to. Because the quantity added to the potential is the box-box potential, and there can be multiple particles within the same box, we also include a check that particle j is the first particle in its box to avoid double-counting the potential. This is easily checked by particle j being in a different box than particle $j-1$. An alternative approach is to use the potential between each particle and the boxes in the interaction list. That would give a more accurate result than using the multipole information for j's box, but the drawback is that we would no longer be able to take advantage of $\Phi(\mathbf{c}_a, \mathbf{c}_b)$ being given by QROM.

The remaining part needed is the direct calculation of the sum over pairwise interactions with particles in neighbouring boxes for $\ell = L$. Note that we do not sum over multiple particles in the same box, because we take level L to correspond to the maximum resolution of the particle positions, so only one electron may occupy the box. The additional sum of the contributions to the potential from particles in the neighbouring boxes at level L is easily accounted for by including neighbouring boxes in the interaction list in Algorithm 5. In this way, this algorithm effectively gives

the quantum implementation of the second pass of the FMM in Algorithm 1, with the quantum implementation of the first pass of Algorithm 1 being given in Algorithm 3.

Accounting for the electron-nuclear potential is more complicated for the adaptive case than for the non-adaptive case. It is no longer possible to choose the nuclei to sum over classically, because the box number is governed by the value in a quantum register, and so the correct nuclei to consider are chosen by that register. A simple way to account for the nuclei is to include registers for the nuclear positions in the list with the electron positions, and similarly associate registers for the multipole information at each level for each nucleus. Because the number of nuclei is no more than the number of electrons in cases of interest, the inclusion of nuclei does not change the asymptotic complexity.

VI. QUANTIFYING THE COMPLEXITY

Now we give the overall complexity of the simulation with the quantum FMM. We will analyse only the adaptive protocol, as it does not require any assumption on the distribution of particles. We first consider the complexity when only using the charge to calculate the potential, then consider the adjustment to the complexity for the multipole information. We also just analyse the complexity for calculating the electron-electron potential, because the nuclei do not change the complexity. In Algorithm 3 for calculating the charge information, there are $L-1$ steps, and for each there are $\eta-1$ steps of copying charge information, as well as η steps of adding charges together. The complexity of the copying or addition is proportional to the number of bits for the charge which is at most $\lceil \log \eta \rceil$. This gives a complexity of

$$\mathcal{O}(\eta \log \eta \log N)\,, \tag{24}$$

where the factor of η is from the iteration over particles, the factor of $\log \eta$ from the arithmetic, and $\log N$ from the number of levels. In addition, the sorting to Morton ordering gives the same contribution to the complexity. This is because sorting η items with optimal sorting networks requires $\mathcal{O}(\eta \log \eta)$ steps. Each step has complexity $\mathcal{O}(\log N)$ because that is the size of the items to be sorted (the particle positions), giving the same complexity as in Eq. (24).

Next we consider the complexity of Algorithm 5 for computing the potential. The contributions to the complexity are as follows.

1. There are $\mathcal{O}(\log N)$ sorts to be performed, with each being a sort of η items and therefore having complexity $\mathcal{O}(\eta \log \eta)$ with optimal sorting networks. The items to be sorted include only the particle positions and the corresponding box charge for level ℓ, and so is of size $\mathcal{O}(\log N)$. Note that we only need the charge information for level ℓ, so do not need to sort the charge information for the other levels at this step. As a result the complexity of the sorts is

$$\mathcal{O}(\eta \log \eta \log^2 N)\,. \tag{25}$$

2. We need to use Algorithm 4 to copy the charge information. This is done $\mathcal{O}(\log N)$ times, and each time there are η steps with $\mathcal{O}(\log \eta)$ information to be copied. As a result the complexity is the same as in Eq. (24).

3. There are $\mathcal{O}(\eta \log N)$ steps where we need to calculate the addition to the potential. There are $\mathcal{O}(1)$ boxes in the interaction list, and so $\Phi(\mathbf{c}_a, \mathbf{c}_b)$ can be determined with a QROM with complexity $\mathcal{O}(1)$ Toffolis, or $\mathcal{O}(\log(1/\epsilon))$ total gates accounting for the precision that the potential is given to. The multiplications by $Q_k(j,\ell)$ and $Q(j,\ell)$ have complexity $\mathcal{O}(\log \eta \log(1/\epsilon))$, and the addition has negligible complexity in comparison. There is also a check whether the information is from a box in the interaction list, which is based on the coordinate and can be performed with $\mathcal{O}(\log N)$ complexity. The error due to the finite size of basis sets scales as $\mathcal{O}(1/N)$ [41–47], so it is reasonable to assume that $\log N = \mathcal{O}(\log(1/\epsilon))$. Then the complexity of the multiplications is larger than any checks of the box. As a result the total complexity of this part is

$$\mathcal{O}(\eta \log \eta \log N \log(1/\epsilon))\,. \tag{26}$$

The complexity in Eq. (26) is as large as that in Eq. (25) (taking $\log N = \mathcal{O}(\log(1/\epsilon))$), and larger than that in Eq. (24), and therefore can be taken as the complexity of the whole procedure.

The leading contributions to the qubits used are as follows.

1. The number of qubits to store the system state is

$$\mathcal{O}(\eta \log N)\,. \tag{27}$$

2. The qubits used for the charge information at all levels is

$$\mathcal{O}(\eta \log \eta \log N) \,. \tag{28}$$

This is because there are $\mathcal{O}(\log N)$ levels, and each uses $\mathcal{O}(\log \eta)$ qubits for the charge.

3. There are temporary qubits used for the coherent sorts. Because these record the results of inequality tests, and there are $\mathcal{O}(\eta \log \eta)$ of these tests, the total number of qubits is

$$\mathcal{O}(\eta \log \eta) \,. \tag{29}$$

The leading contribution to the logical qubits used is $\mathcal{O}(\eta \log \eta \log N)$.

More generally, for the multipole expansion, the order needed for precision ϵ is $\mathcal{O}(\log(1/\epsilon))$. The number of multipole moments scales as the square of the order, and each needs $\mathcal{O}(\log(1/\epsilon))$ bits. As a result, rather than just the charge being given with $\lceil \log \eta \rceil$ bits, the multipole information needs a number of bits $\mathcal{O}(\log^3(1/\epsilon))$. The number of qubits used can therefore be given as

$$\mathcal{O}(\eta \log N \log^3(1/\epsilon)) \,. \tag{30}$$

Similarly, for the calculation of the multipole information rather than charge for each box, the factor of $\log \eta$ can be replaced with $\mathrm{polylog}(1/\epsilon)$, giving the complexity of calculating the multipole information for boxes as

$$\mathcal{O}(\eta \log N \,\mathrm{polylog}(1/\epsilon)) \,. \tag{31}$$

Then for the calculation of the potential, there is a similar complexity, because there are $\mathcal{O}(\eta \log N)$ steps where the multipole information needs to be used to calculate the contribution to the potential.

For the overall complexity, we need to account for the number of Trotter steps as well. This is accounted for in Ref. [17], which we summarise here. A real-space grid Hamiltonian may be given as

$$H = \sum_{j,k,\sigma} \tau_{j,k} a^\dagger_{j,\sigma} a_{k,\sigma} + \sum_{l,m,\sigma,\tau} \nu_{l,m} a^\dagger_{l,\sigma} a_{l,\sigma} a^\dagger_{m,\tau} a_{m,\tau} \,, \tag{32}$$

where $a^\dagger_{j,\sigma}$ and $a_{j,\sigma}$ are creation and annihilation operators, $\{j,k,l,m\}$ are orbital indices, and $\{\sigma,\tau\}$ are spin indices. The spectral norm error in a fixed number particle manifold for an order-k product formula $S_k(t)$ can be estimated as [17]

$$\left\| S_k(t) - e^{-itH} \right\|_{W_\eta} = \mathcal{O}\left(\left(\|\nu\|_{1,[\eta]} + \|\tau\|_1 \right)^{k-1} \|\tau\|_1 \, \|\nu\|_{1,[\eta]} \, \eta \, t^{k+1} \right) \,, \tag{33}$$

where the norms are defined as

$$\|\nu\|_{1,[\eta]} = \max_j \max_{k_1 < \ldots < k_\eta} \left(|v_{j,k_1}| + \ldots + |v_{j,k_\eta}| \right) , \tag{34}$$

$$\|\tau\|_1 = \max_j \sum_k |\tau_{j,k}| \,. \tag{35}$$

The number of time steps needed for a product formula of order k is then

$$\mathcal{O}\left(t^{1+1/k} (\|\nu\|_{1,[\eta]} + \|\tau\|_1)^{1-1/k} (\|\tau\|_1 \, \|\nu\|_{1,[\eta]} \, \eta/\epsilon)^{1/k} \right) \,. \tag{36}$$

The scaling of the norms is

$$\|\nu\|_{1,[\eta]} = \mathcal{O}\left(\frac{\eta^{2/3} N^{1/3}}{\Omega^{1/3}} \right) , \tag{37}$$

$$\|\tau\|_1 = \mathcal{O}\left(\frac{N^{2/3}}{\Omega^{2/3}} \right) . \tag{38}$$

This then gives the number of steps as

$$\mathcal{O}\left(t^{1+1/k} \left(\frac{\eta^{2/3} N^{1/3}}{\Omega^{1/3}} + \frac{N^{2/3}}{\Omega^{2/3}} \right)^{1-1/k} \left(\frac{\eta^{5/3} N}{\Omega \epsilon} \right)^{1/k} \right) . \tag{39}$$

Since k may be chosen arbitrarily large (though in practice a moderate value will be optimal), we can give the complexity with $1/k$ replaced by $o(1)$, which gives

$$t\left(\frac{\eta^{2/3}N^{1/3}}{\Omega^{1/3}}+\frac{N^{2/3}}{\Omega^{2/3}}\right)\left(\frac{\eta N t}{\Omega\epsilon}\right)^{o(1)}. \tag{40}$$

The $\mathcal{O}$ is not given for the entire expression, because the fact that this is an asymptotic scaling is accounted for in the $o(1)$. The factors of $\log N$ and $\mathrm{polylog}(1/\epsilon)$ for the complexity of the FMM calculation can be included with the $N^{o(1)}$ and $1/\epsilon^{o(1)}$. Therefore, multiplying by the Toffoli complexity of the FMM calculation gives the Toffoli complexity of the complete simulation as

$$t\left(\frac{\eta^{5/3}N^{1/3}}{\Omega^{1/3}}+\frac{\eta N^{2/3}}{\Omega^{2/3}}\right)\left(\frac{\eta N t}{\Omega\epsilon}\right)^{o(1)}, \tag{41}$$

as given above in Eq. (7). A common assumption to simplify the complexity is the thermodynamic limit, where $\eta \propto \Omega$. This gives the complexity as

$$t\left(\eta^{4/3}N^{1/3}+\eta^{1/3}N^{2/3}\right)\left(\frac{\eta N t}{\epsilon}\right)^{o(1)}, \tag{42}$$

which is the complexity stated in the abstract. In this expression $\eta^{o(1)}$ is included because there is a factor of $(\eta^{5/3}/\Omega)^{1/k}$ in Eq. (39), which gives $\eta^{2/3k}$ for $\eta \propto \Omega$.

VII. CONCLUSION

The need to perform a double sum with complexity $\mathcal{O}(\eta^2)$ to compute the potential energy has been a long-standing bottleneck in simulation of quantum chemistry based on product formulae. The fast multipole method provides a speedup to $\widetilde{\mathcal{O}}(\eta)$ in classical computing, but the difficulty in translating the method to quantum algorithms is that it requires data accesses at positions according to the values in data registers, particularly for the adaptive case. That would negate the speedup due to the overhead in quantum data access.

Here, we have provided a method to overcome this problem, yielding a quantum algorithm for FMM with complexity close to linear in the number of particles η. Our method uses a similar principle as the quantum sort, where the data accesses are made in a fixed sequence of locations. The procedure is considerably more difficult than the quantum sort, because it is also necessary to access data from boxes in the interaction list.

To overcome that problem, we combined the quantum sort with a set of shifted Morton orderings, in order to provide a moderate set of sequences which enable access to all boxes in the interaction list. To access this information we associated box information with each particle, and used a procedure for copying the data from neighbouring boxes.

The result of this advance is that the overall complexity scaling is better than in prior work. For grid-based approaches, large numbers of grid points (or plane waves) are required in order to provide accurate simulation, such that $N \gg \eta$. In this limit, the best complexity is achieved by using interaction picture simulation, which provides a $N^{1/3}$ scaling [33]. That method however has an additional factor of η^3, and so the method presented here has the best complexity for $N < \eta^6$, which is the typical range considered for realistic examples. For more comparable methods, such as the first-quantised product formula approach in Ref. [4], we improve over the complexity by a factor of η (up to logarithmic factors).

There are still potential improvements to be made in the efficiency of the procedure. For example, redundant information for all boxes is included by associating box information for each electron register. It is possible that further improvements could be made by using only a single copy of information for each box (a principle used in Ref. [15]). There is also redundancy in our usage of multiple Morton orderings, which results in the information from boxes in the interaction list being accessed multiple times. Potentially, alternatives such as Hilbert orderings [38, 39] or different shifts in the Morton ordering [40] could provide better performance.

In the future it may also be interesting to use our scheme as a basis for efficient implementations of the split-operator method for Hamiltonians with more general potentials. The fast multipole method is related to the method of hierarchical matrices [48], which allows matrix-vector multiplication of general kernel matrices $\Phi(\|\mathbf{r}_i - \mathbf{r}_j\|)$ of dimension n in time $\mathcal{O}(n\,\mathrm{polylog}(n))$. While related, the fast multipole does not straightforwardly map onto these more general methods. In particular, hierarchical matrix methods replace the interaction list with *admissible blocks*, which are defined by well-separated pairs of clusters of particles. However, these clusters are not confined to one level of the hierarchical splitting of the kernel and so require additional arithmetic to avoid overlapping blocks. The interaction list of a box in our fast multipole scheme can therefore be understood as a subset of the set of admissible blocks. Future work is required to include such methods in the context of split-operator-based simulation.

ACKNOWLEDGMENTS

The authors thank Guang Hao Low, Nathan Wiebe, Rolando Somma, Robin Kothari, and Pedro Costa for helpful discussion. DWB worked on this project under a sponsored research agreement with Google Quantum AI. DWB is also supported by Australian Research Council Discovery Project DP220101602. ADB was supported by the DOE Office of Fusion Energy Sciences "Foundations for quantum simulation of warm dense matter" project. ECE and AT are supported by Google Quantum AI. This article has been co-authored by employees of National Technology & Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The authors own all right, title and interest in and to the article and are solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan.

---

[1] Y. Su, D. W. Berry, N. Wiebe, N. Rubin, and R. Babbush, Fault-tolerant quantum simulations of chemistry in first quantization, PRX Quantum **2**, 040332 (2021).

[2] D. W. Berry, N. C. Rubin, A. O. Elnabawy, G. Ahlers, A. E. DePrince III, J. Lee, C. Gogolin, and R. Babbush, Quantum simulation of realistic materials in first quantization using non-local pseudopotentials, npj Quantum Information **10**, 130 (2024).

[3] R. Babbush, W. J. Huggins, D. W. Berry, S. F. Ung, A. Zhao, D. R. Reichman, H. Neven, A. D. Baczewski, and J. Lee, Quantum simulation of exact electron dynamics can be more efficient than classical mean-field methods, Nature Communications **14**, 4058 (2023).

[4] N. C. Rubin, D. W. Berry, A. Kononov, F. D. Malone, T. Khattar, A. White, J. Lee, H. Neven, R. Babbush, and A. D. Baczewski, Quantum computation of stopping power for inertial fusion target design, Proceedings of the National Academy of Sciences **121**, e2317772121 (2024).

[5] S. Lloyd, Universal Quantum Simulators, Science **273**, 1073 (1996).

[6] N. Wiebe and A. M. Childs, Hamiltonian simulation using linear combinations of unitary operations, Quantum Information and Computation **12**, 901 (2012).

[7] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, Exponential improvement in precision for simulating sparse Hamiltonians, in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14 (ACM, New York, NY, USA, 2014) pp. 283–292.

[8] D. W. Berry and A. M. Childs, Black-box Hamiltonian simulation and unitary implementation, Quantum Information and Computation **12**, 0029 (2012).

[9] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, Simulating Hamiltonian dynamics with a truncated Taylor series, Phys. Rev. Lett. **114**, 090502 (2015).

[10] G. H. Low and I. L. Chuang, Optimal Hamiltonian simulation by quantum signal processing, Phys. Rev. Lett. **118**, 010501 (2017).

[11] G. H. Low and I. L. Chuang, Hamiltonian Simulation by Qubitization, Quantum **3**, 163 (2019).

[12] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu, Theory of Trotter error with commutator scaling, Phys. Rev. X **11**, 011020 (2021).

[13] M. E. S. Morales, P. C. S. Costa, G. Pantaleoni, D. K. Burgarth, Y. R. Sanders, and D. W. Berry, Selection and improvement of product formulae for best performance of quantum simulation, Quantum Information & Computation **25**, 1 (2025).

[14] G. H. Low and N. Wiebe, Hamiltonian simulation in the interaction picture, arXiv:1805.00675 (2019).

[15] T. Stetina and N. Wiebe, First-quantized quantum simulation of non-relativistic QED with emergent topologically protected Coulomb interactions, arXiv:2508.19343 (2025).

[16] I. Kassal, S. P. Jordan, P. J. Love, M. Mohseni, and A. Aspuru-Guzik, Polynomial-time quantum algorithm for the simulation of chemical dynamics, Proceedings of the National Academy of Sciences **105**, 18681 (2008).

[17] G. H. Low, Y. Su, Y. Tong, and M. C. Tran, Complexity of implementing Trotter steps, PRX Quantum **4**, 020323 (2023).

[18] V. Rokhlin, Rapid solution of integral equations of classical potential theory, Journal of Computational Physics **60**, 187 (1985).

[19] J. Barnes and P. Hut, A hierarchical O(N log N) force-calculation algorithm, Nature **324**, 446–449 (1986).

[20] J. Carrier, L. Greengard, and V. Rokhlin, A fast adaptive multipole algorithm for particle simulations, SIAM Journal on Scientific and Statistical Computing **9**, 669 (1988).

[21] A. M. Childs, J. Leng, T. Li, J.-P. Liu, and C. Zhang, Quantum simulation of real-space dynamics, Quantum **6**, 860 (2022).

[22] S.-T. Cheng and C.-Y. Wang, Quantum switching and quantum merge sorting, IEEE Transactions on Circuits and Systems I: Regular Papers **53**, 316 (2006).

[23] R. Beals, S. Brierley, O. Gray, A. W. Harrow, S. Kutin, N. Linden, D. Shepherd, and M. Stather, Efficient distributed quantum computing, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **469**, 20120686

(2013).
[24] A. Yesypenko, C. Chen, and P.-G. Martinsson, A simplified fast multipole method based on strong recursive skeletonization, Journal of Computational Physics **524**, 113707 (2025).
[25] R. E. Angulo and O. Hahn, Large-scale dark matter simulations, Living Reviews in Computational Astrophysics **8**, 1 (2022).
[26] I. Fukuda and H. Nakamura, Non-Ewald methods for evaluating the electrostatic interactions of charge systems: similarity and difference, Biophysical Reviews **14**, 1315–1340 (2022).
[27] Q. Wang, A hybrid Fast Multipole Method for cosmological n-body simulations, Research in Astronomy and Astrophysics **21**, 003 (2021).
[28] N. Y. Gnedin, Hierarchical particle mesh: An FFT-accelerated fast multipole method, The Astrophysical Journal Supplement Series **243**, 19 (2019).
[29] B. Stenqvist, V. Aspelin, and M. Lund, Generalized moment correction for long-ranged electrostatics, Journal of Chemical Theory and Computation **16**, 3737 (2020).
[30] J. Liang, L. Lu, A. Barnett, L. Greengard, and S. Jiang, Accelerating fast Ewald summation with prolates for molecular dynamics simulations, arXiv:2505.09727 (2025).
[31] Y. Su, H.-Y. Huang, and E. T. Campbell, Nearly tight Trotterization of interacting electrons, Quantum **5**, 495 (2021).
[32] G. H. Low and N. Wiebe, Hamiltonian Simulation in the Interaction Picture, arXiv:1805.00675 (2018).
[33] R. Babbush, D. W. Berry, J. R. McClean, and H. Neven, Quantum Simulation of Chemistry with Sublinear Scaling in Basis Size, npj Quantum Information **5**, 92 (2019).
[34] R. Babbush, N. Wiebe, J. McClean, J. McClain, H. Neven, and G. K.-L. Chan, Low-Depth Quantum Simulation of Materials, Physical Review X **8**, 011044 (2018).
[35] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, Encoding electronic spectra in quantum circuits with linear T complexity, Physical Review X **8**, 041015 (2018).
[36] I. D. Kivlichan, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, W. Sun, Z. Jiang, N. Rubin, A. Fowler, A. Aspuru-Guzik, H. Neven, and R. Babbush, Improved Fault-Tolerant Quantum Simulation of Condensed-Phase Correlated Electrons via Trotterization, Quantum **4**, 296 (2020).
[37] T. Häner, M. Roetteler, and K. M. Svore, Optimizing Quantum Circuits for Arithmetic, arXiv:1805.12445 (2018).
[38] S. Liao, M. Lopez, and S. Leutenegger, High dimensional similarity search with space filling curves, in *Proceedings 17th International Conference on Data Engineering* (2001) pp. 615–622.
[39] K. Barkalov, A. Shtanyuk, and A. Sysoyev, A fast kNN algorithm using multiple space-filling curves, Entropy **24**, 767 (2022).
[40] S. Li, L. Simons, J. B. Pakaravoor, F. Abbasinejad, J. D. Owens, and N. Amenta, kANN on the GPU with shifted sorting, in *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*, EGGH-HPG'12 (Eurographics Association, Goslar, DEU, 2012) p. 39–47.
[41] A. Grüneis, J. J. Shepherd, A. Alavi, D. P. Tew, and G. H. Booth, Explicitly correlated plane waves: Accelerating convergence in periodic wavefunction expansions, The Journal of Chemical Physics **139**, 84112 (2013).
[42] C. Hättig, W. Klopper, A. Köhn, and D. P. Tew, Explicitly correlated electrons in molecules, Chemical Reviews **112**, 4 (2011).
[43] J. Harl and G. Kresse, Cohesive energy curves for noble gas solids calculated by adiabatic connection fluctuation-dissipation theory, Physical Review B **77**, 45136 (2008).
[44] J. J. Shepherd, A. Grüneis, G. H. Booth, G. Kresse, and A. Alavi, Convergence of many-body wave-function expansions using a plane-wave basis: From homogeneous electron gas to solid state systems, Physical Review B **86**, 35111 (2012).
[45] T. Helgaker, W. Klopper, H. Koch, and J. Noga, Basis-set convergence of correlated calculations on water, Journal of Chemical Physics **106**, 9639 (1998).
[46] W. Klopper, Limiting values for Møller-Plesset second-order correlation energies of polyatomic systems: A benchmark study on Ne, HF, H_2O, N_2, and He...He, The Journal of Chemical Physics **102**, 6168 (1995).
[47] A. Halkier, T. Helgaker, P. Jørgensen, W. Klopper, H. Koch, J. Olsen, and A. K. Wilson, Basis-set convergence in correlated calculations on Ne, N_2, and H_2, Chemical Physics Letters **286**, 243 (1998).
[48] M. Fenn and G. Steidl, FMM and H-matrices: A short introduction to the basic idea (2002).
[49] L. Greengard and V. Rokhlin, *On the Efficient Implementation of the Fast Multipole Algorithm*, Tech. Rep. (Yale University Department of Computer Science, 1988).
[50] L. Greengard and V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, Acta Numerica **6**, 229 (1997).
[51] B. Shanker and H. Huang, Accelerated Cartesian expansions–A fast method for computing of potentials of the form $R^{-\nu}$ for all real ν, Journal of Computational Physics **226**, 732 (2007).

Appendix A: Additional FMM implementation details

The main text primarily considers a simplified version of FMM that uses zeroth-order expansions (see Algorithm 1). However, higher-order expansions are essential to achieving arbitrary accuracy in approximating V. In this Appendix, we consider some of the details needed to extend the results in the main text to higher order.

All versions of FMM make use of the fact that the far-field (source-free) contribution to the potential satisfies the

Laplace equation. A generic solution to the Laplace equation can be written as

$$\sum_{l=0}^{\infty} \sum_{m=-l}^{l} \left(L_{lm} \|\mathbf{r}_o - \mathbf{r}_s\|^l + M_{lm} \|\mathbf{r}_o - \mathbf{r}_s\|^{-l-1} \right) Y_{lm}(\theta_{so}, \phi_{so}), \tag{A1}$$

where $\mathbf{r}_s$ ($\mathbf{r}_o$) is the centroid of a box containing a distribution of source (observer) particles, θ_{so} (ϕ_{so}) is the polar (azimuthal) angle of the axis of separation between these boxes, and Y_{lm} are the spherical harmonics. An expansion of the form in Eq. (A1) in which $L_{lm} = 0$ ($M_{lm} = 0$) is a multipole (local) expansion. In practice, either type of expansion is truncated beyond some fixed order $l = \mathcal{P}$ and thus comprised of $(\mathcal{P}+1)^2$ terms.

Everything in the main text is a special case in which $\mathcal{P} = 0$. In this special case, the multipole expansion coefficient M_{00} for each box is the total charge in that box, and the local expansion coefficient L_{00} for each box is the average far-field contribution to the potential. Higher-order implementations require machinery for storing and manipulating higher moments of the charge distribution and potential.

Algorithm 6 provides the details of an order-$\mathcal{P}$ implementation of FMM [49]. Relative to Algorithm 1, the new components of this implementation include

1. $(\mathcal{P}+1)^2$ multipole and local expansion coefficients for each box outside of the leaf level,
2. operators ($\mathcal{T}_{MM}$ and $\mathcal{T}_{LL}$) that translate the origins of the multipole and local expansions (respectively) from the centroid of one box to the centroid of another, and
3. operators ($\mathcal{T}_{ML}$) that translate the multipole expansion coefficients about one box into local expansion coefficients about a box in its interaction list.

The precise details of the various translation operators are one of the primary considerations in an FMM implementation, and they determine the cost of tree traversal. Generically, they only depend on the structure of the tree and the form of the potential (Φ), so they do not require any quantum data access and can be classically precomputed.

In the adaptive version of the quantum FMM in the main text, there is a single grid point per leaf-level box. Thus, there is no need for additional leaf-level storage for order-$\mathcal{P}$ quantum FMM because the monopole moments are the only non-zero moments. However, other levels of the tree will require $\mathcal{O}((\mathcal{P}+1)^2 \log(1/\epsilon))$ qubits per box to store all of the multipole and local expansion coefficients. The M2M and L2L translation operators (see loops 2 and 3 of Algorithm 6, respectively) will be applied to generate these coefficients. Each translation is in 1 of 8 directions to the centroid of its parent (or child), with the distances only depending on which of the $L-1$ pairs of adjacent levels are being traversed. Similarly, there are at most 189 unique M2L translations (see loop 3 of Algorithm 6) in any of the $L-1$ levels in which those translations take place. An 8-fold cubic symmetry might also be exploited in implementations in which the M2L translation operators strictly depend on the distance between centroids (i.e., any translation can be mapped to an equivalent one in the first octant).

One of the key design choices in an FMM implementation is the manner in which these translation operators are applied. A naïve classical implementation would require $\widetilde{\mathcal{O}}(\mathcal{P}^4)$ floating-point operations per translation for dense translation operators. Early on in the history of FMM, it was realised that translation operators applied using fast Fourier transforms or in a basis in which they're diagonal would only require $\widetilde{\mathcal{O}}(\mathcal{P}^2)$ operations [49]. Achieving this scaling in practice required mitigating certain numerical instabilities at the cost of additional implementation complexity [50]. Alternatively, formulations based on Cartesian tensors would require $\widetilde{\mathcal{O}}(\mathcal{P}^6)$ operations for translation, but could benefit from other design trade-offs (e.g., exact M2M and L2L translation could be used to reduce the effective size of the interaction list and the number of M2L translations) [51]. It also warrants noting that certain formulations might require complex arithmetic at intermediate levels, even if the resulting potentials are real. Determining an approach to translation that is best suited to a practical quantum implementation is left to future work.

Finally, we comment on how the choice of $\mathcal{P}$ impacts the overall approximation error in V. The approximation error in truncating $V_{i,\text{far}}$ beyond order $\mathcal{P}$ is bounded from above as

$$|V_{i,\text{far}}(\mathcal{P} \to \infty) - V_{i,\text{far}}(\mathcal{P})| \leq \frac{g}{\Omega^{1/3}} \left(\frac{1}{2^{\mathcal{P}-L+3}} \right), \tag{A2}$$

where this bound comes from a routine application of the triangle inequality between leaf-level boxes in each other's interaction lists (i.e., their centroids are separated by at least $\Omega^{1/3}/2^{L-2}$) and g is some $\mathcal{O}(1)$ constant that depends on the underlying charge distribution. Thus, it is evident that $\mathcal{P}$ should be $\mathcal{O}(\log(1/\epsilon))$ to ensure an overall approximation error that is $\mathcal{O}(\epsilon)$.

Algorithm 6: Classical FMM (order $\mathcal{P}$)

Initialise $V \leftarrow 0$
Upward pass
1. **for** $b \in B_L$
  for $l \leftarrow 0$ **to** $\mathcal{P}$
    for $m \leftarrow -l$ **to** l
      Initialise $M_{lm}(b) \leftarrow 0$
      for $j \in p(b)$
        $M_{lm}(b) \leftarrow M_{lm}(b) + q_j \left\|\mathbf{r}_j - \mathbf{c}_b\right\|^l Y_l^{-m}(\theta_j, \phi_j)$
      end
    end
  end
end
2. **for** $\ell \leftarrow L-1$ **to** 3
  for $b \in B_\ell$
    for $l \leftarrow 0$ **to** $\mathcal{P}$
      for $m \leftarrow -l$ **to** l
        Initialise $M_{lm}(b) \leftarrow 0$
        for $b' \in C(b)$
          $M_{lm}(b) \leftarrow M_{lm}(b) + \mathcal{T}_{MM}(b, b') M_{lm}(b')$
        end
      end
    end
  end
end
Downward pass
Initialise $L_{lm}(0) \leftarrow 0$
3. **for** $\ell \leftarrow 2$ **to** L
  for $b \in B_\ell$
    for $l \leftarrow 0$ **to** $\mathcal{P}$
      for $m \leftarrow -l$ **to** l
        Initialise $L_{lm}(b) \leftarrow 0$
        for $b' \in P(b)$
          $L_{lm}(b) \leftarrow L_{lm}(b) + \mathcal{T}_{LL}(b, b') L_{lm}(b')$
        end
        for $b' \in I(b)$
          $L_{lm}(b) \leftarrow L_{lm}(b) + \mathcal{T}_{ML}(b, b') M_{lm}(b')$
        end
      end
    end
  end
end
4. **for** $b \in B_L$
  for $i \in p(b)$
    Initialise $V_i \leftarrow 0$
    for $l \leftarrow 0$ **to** $\mathcal{P}$
      for $m \leftarrow -l$ **to** l
        $V_i \leftarrow V_i + \left\|\mathbf{r}_i - \mathbf{c}_b\right\|^l Y_l^m(\theta_i, \phi_i)$
      end
    end
    for $n \in NN(b)$
      for $j \in p(n)$, $i > j$
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j$
      end
    end
    for $j \in p(b)$, $i > j$
      $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j$
    end
  end
  $V \leftarrow V + q_i V_i$
end
Return V