

Quantum Simulation of Electronic Structure via Quantum Fast Multipole Method

Dominic W. Berry^{1,*} Kianna Wan,^{2,3} Andrew D. Baczewski⁴,
 Elliot C. Eklund⁵, Arkin Tikku⁵ and Ryan Babbush^{2,†}

¹*School of Mathematical and Physical Sciences, Macquarie University, New South Wales 2109, Australia*

²*Google Quantum AI, Venice, California 90291, USA*

³*Stanford Institute for Theoretical Physics, Stanford University, Stanford, California 94305, USA*

⁴*Quantum Algorithms and Applications Collaboratory, Sandia National Laboratories, Albuquerque, New Mexico 87185, USA*

⁵*School of Chemistry, University of Sydney, NSW 2006, New South Wales, Australia*



(Received 8 December 2025; revised 8 April 2026; accepted 1 May 2026; published 10 August 2026)

Here we describe an approach for simulating electronic structure on quantum computers with significantly lower asymptotic complexity than prior work. The approach uses a real-space first-quantized representation of the molecular Hamiltonian, which we propagate using high-order product formulas. Essential for this low complexity is the use of a technique similar to the fast multipole method for computing the Coulomb operator with $\tilde{\mathcal{O}}(\eta)$ 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 $N < \eta^7$ (the regime of practical interest), with only first-quantized interaction-picture simulations providing better performance for $N > \eta^7$. As with the classical fast multipole method, large particle numbers $\eta \gtrsim 10^3$ would be needed to realize this advantage.

DOI: 10.1103/b2vn-zf6c

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 [1]. 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 optimizing algorithms for quantum simulation. Even more broadly, simulations of materials [2,3] and degenerate plasmas [4,5] 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 formulas [6]. As the field of quantum algorithms matured, more advanced methods were introduced. These algorithms included techniques based on linear combinations of unitaries [7,8], quantum walks [9], Taylor series [10], quantum signal processing [11], and block encoding [12]. A key breakthrough with these more recent methods is that they provide a relatively favorable complexity scaling in the simulation error ϵ , scaling as $\mathcal{O}(\log(1/\epsilon))$. This is in contrast to product formulas, which scale as $\text{poly}(1/\epsilon)$. However, recent work shows that product formulas can be competitive with logarithmically scaling methods for simulation because they provide low error in practice [13,14].

Simulating the types of Hamiltonians that are relevant in chemistry is made difficult by the fact that all pairwise

* Contact author: dominic.berry@mq.edu.au

† Contact author: ryanbabbush@gmail.com

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International license](#). Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

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-quantized 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.

For second quantization, the Coulomb potential may be calculated with a fast Fourier transform, avoiding the double sum [15]. However, due to second quantization the complexity is at least linear in the number of orbitals N . In cases where $N \gg \eta$, it is preferable to use first-quantized-based approaches to avoid the factor of N in the complexity. We note that very recently, a first-quantized approach for simulating non-relativistic quantum electrodynamics was proposed that also avoids the $\mathcal{O}(\eta^2)$ overhead for the Coulomb potential [16].

Simulations based on product formulas in first quantization [5,17] show promise for better scaling than many of these algorithms, due to the Low *et al.* [18] 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, tree codes (e.g., Barnes-Hut [19]) and the fast multipole method (FMM) [20,21] enable computation of the potential to be performed with complexity $\mathcal{O}(\eta \text{polylog}(\eta) \text{polylog}(1/\epsilon))$ or $\mathcal{O}(\eta \text{polylog}(1/\epsilon))$, respectively. The difficulty with applying these methods in quantum algorithms is that a direct translation of the various classical algorithms 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 either tree codes or FMMs. The FMM approach discussed in Ref. [22], 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 quantum random access memory (QRAM). A detailed discussion of the issues associated with implementing an FMM quantum algorithm is provided by Babbush *et al.* in Ref. [4]. (Note that the FMM is only relevant to first quantization, because for second quantization the approach of Ref. [15] may be used.)

Our solution to recover the $\mathcal{O}(\eta \log \eta)$ scaling of the classical algorithm is related to the implementation of a quantum sort [23,24] 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 [25–31].

The fast multipole method needs large particle numbers in order to show an improvement over direct summation. Greengard and Rokhlin show a break-even point of about 500 particles for modest accuracy to 5000 particles for high-accuracy calculations [32]. In comparison, Ref. [5] involves 1729 electrons for simulation of a deuterium plasma. Simulation of plasma with larger atomic numbers can be expected to require simulation of thousands of electrons, where the quantum fast multipole method becomes relevant. We caution that the current quantum implementation of the FMM has large overheads and will require considerable further optimization to provide comparable performance to the classical FMM. The focus of this work is on showing that the same scaling as the classical FMM is possible.

In the rest of this paper, we summarize the electronic structure Hamiltonian in Sec. II, and the FMM algorithm in Sec. III. In Sec. 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 Sec. 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 Sec. VI and conclude in Sec. VII.

II. SIMULATING QUANTUM CHEMISTRY IN REAL-SPACE FIRST QUANTIZATION

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

$$H = T + U + V + \sum_{l \neq k=1}^{\zeta} \frac{q_l q_k}{2 \|R_l - R_k\|}, \quad (1)$$

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

$$U = - \sum_{i=1}^{\eta} \sum_{l=1}^{\zeta} \sum_{\mathbf{p} \in G} \frac{q_l}{\|R_l - \mathbf{r}_{\mathbf{p}}\|} |\mathbf{p}\rangle \langle \mathbf{p}|_i, \quad (3)$$

$$V = \sum_{i \neq j=1}^{\eta} \sum_{\mathbf{p}, \mathbf{q} \in G} \frac{1}{2 \|\mathbf{r}_{\mathbf{p}} - \mathbf{r}_{\mathbf{q}}\|} |\mathbf{p}\rangle \langle \mathbf{p}|_i |\mathbf{q}\rangle \langle \mathbf{q}|_j, \quad (4)$$

where QFT_i denotes the standard quantum Fourier transform applied to register i and $\|\cdot\|$ denotes the Euclidean norm. Further, the indices l and κ label the classical data for the nuclei, while i and j label the electronic registers. 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 $\hbar$, $4\pi\epsilon_0$, 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

$$\begin{aligned} \mathbf{r}_p &= \frac{\mathbf{p}\Omega^{1/3}}{N^{1/3}}, & \mathbf{k}_p &= \frac{2\pi\mathbf{p}}{\Omega^{1/3}}, \\ \mathbf{p} &\in G, & G &= \left[-\frac{N^{1/3}-1}{2}, \frac{N^{1/3}-1}{2} \right]^3. \end{aligned} \quad (5)$$

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.* [17] in the first work on quantum simulating chemistry in first quantization. Our approach is then to perform simulation under the electronic structure Hamiltonian by using high-order product formulas 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.* [17], resulting in $\tilde{\mathcal{O}}(\eta)$ complexity. In cases where ζ is small, it would be appropriate to perform a simulation under the operator U using the approach from Kassal *et al.* [17] 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.* [17], we propose to perform time evolution using high-order product formulas. The work of Low *et al.* [18] bounds the number of Trotter steps required for a split-operator approach in real-space second quantization. This result for the number of Trotter steps also holds for real-space first quantization (see the Supporting Information of Ref. [5], Sec. V.B.), and is

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

for time-evolution duration t and target precision ϵ . 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.* [17], has complexity $\tilde{\mathcal{O}}(\eta^2)$. But we will instead show an approach based on the fast multipole method with complexity $\tilde{\mathcal{O}}(\eta)$. Multiplying the number of Trotter steps by $\tilde{\mathcal{O}}(\eta)$, this will lead to an overall gate complexity

$$\begin{aligned} & t \left(\frac{\eta^{5/3} N^{1/3}}{\Omega^{1/3}} + \frac{\eta N^{2/3}}{\Omega^{2/3}} \right) \left(\frac{\eta N t}{\Omega \epsilon} \right)^{o(1)} \\ &= t (\eta^{4/3} N^{1/3} + \eta^{1/3} N^{2/3}) \left(\frac{\eta N t}{\epsilon} \right)^{o(1)}, \end{aligned} \quad (7)$$

where the right-hand side is a simplified expression if $\Omega \propto \eta$ (the thermodynamic limit). The space complexity is $\mathcal{O}(\eta \log N \text{ polylog}(1/\epsilon))$ 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-quantized plane wave algorithms developed by Su *et al.* [33] and Low and Wiebe [15]. 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-quantized interaction-picture method [34] but that has much larger scaling with η .

To compare these complexities more precisely, we also take into account the volume Ω . The commonly considered case is the thermodynamic limit, where Ω is increased proportional to η . In this limit, the result using the multipole approach has better complexity than that of the interaction picture method in Ref. [34] given $N < \eta^7$. (If Ω were constant, then our method would be best for $N < \eta^6$.) As a result, the result presented here has the lowest known complexity for any approach whenever $N < \eta^7$, which is the result quoted in the abstract. See Box 1 below for a summary of the method and resulting complexity.

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

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 the number of basis functions, $\eta < N$ is the number of electrons, Ω is the computational cell volume, and ϵ is the target precision.

Year	References	Primary innovation	Toffoli/T complexity
2017	Babbush <i>et al.</i> [35]	Using plane waves with Trotter	$\tilde{\mathcal{O}}(\eta^2 N^{17/6} \sqrt{1 + \eta \Omega^{1/3} / N^{1/3}} / (\Omega^{5/6} \epsilon^{3/2}))$
2017	Babbush <i>et al.</i> [35]	Using plane waves with LCU	$\tilde{\mathcal{O}}((N^4 / \Omega^{1/3} + N^{11/3} / \Omega^{2/3}) / \epsilon)$
2018	Babbush <i>et al.</i> [36]	Linear scaling quantum walks	$\tilde{\mathcal{O}}((N^{10/3} / \Omega^{1/3} + N^{8/3} / \Omega^{2/3}) / \epsilon)$
2018	Low and Wiebe [15]	Interaction picture with second quantization	$\tilde{\mathcal{O}}(N^{8/3} / (\Omega^{2/3} \epsilon))$
2018	Babbush <i>et al.</i> [34]	First quantized qubitisation	$\tilde{\mathcal{O}}((\eta^3 (N/\Omega)^{1/3} + \eta^2 (N/\Omega)^{2/3}) / \epsilon)$
2018	Babbush <i>et al.</i> [34]	Interaction picture with first quantization	$\tilde{\mathcal{O}}(\eta^3 (N/\Omega)^{1/3} / \epsilon)$
2019	Kivlichan <i>et al.</i> [37]	Better Trotter steps	$\tilde{\mathcal{O}}(N^3 / (\Omega^{2/3} \epsilon^{3/2}))$
2019	Childs <i>et al.</i> [13]	Tighter Trotter bounds	$N^{7/3+o(1)} / (\Omega^{1/3} \epsilon^{1+o(1)})$
2021	Su <i>et al.</i> [33]	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 <i>et al.</i> [18]	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 <i>et al.</i> [5]	Tighter Trotter bounds in first quantization	$\eta^2 (\eta^{2/3} (N/\Omega)^{1/3} + (N/\Omega)^{2/3}) (\eta N)^{o(1)} / \epsilon^{1+o(1)}$
2025	Stetina and Wiebe [16]	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	$\eta (\eta^{2/3} (N/\Omega)^{1/3} + (N/\Omega)^{2/3}) (\eta N)^{o(1)} / \epsilon^{1+o(1)}$

distributed within a cubic simulation cell. Note that our quantum implementation does not rely on uniform distribution; in Sec. V we describe the full adaptive algorithm. Let q_i and $\mathbf{r}_i$ ($i \in [\eta] := \{1, \dots, \eta\}$) be the charge and position of the i th particle, respectively. Our objective is

to compute the total potential energy given by

$$V = \frac{1}{2} \sum_{i \in [\eta]} \sum_{\substack{j \in [\eta] \\ j \neq i}} q_i \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j = \sum_{i \in [\eta]} q_i V_i, \quad (10)$$

Box 1: Summary of methods and results

- (1) Real-space first quantization is used with N grid points for η electrons.
- (2) Evolution for time t with allowable error ϵ is simulated using high-order product formulas with the number of steps given in Eq. (6).
- (3) The exponentials of the kinetic energy and potential are simulated by calculating these energies in the computational basis and applying phase factors.
- (4) The potential energy is calculated using a quantum form of the FMM in either one of two methods.
 - (a) A non-adaptive method with a fixed grid (see Sec. IV). The principle is that the electrons are moved into registers corresponding to the boxes for the FMM via an approach based on a quantum sort (see Algorithm 3). This method can only act on the subspace where the electrons are evenly distributed.
 - (b) An adaptive method that can account for an arbitrary distribution of particles (see Sec. V). This works via three main principles.
 - (i) The multipole information for each box is associated with a register for an electron and is moved as the registers are sorted.
 - (ii) The information from neighboring boxes is obtained by copying it along a sorted list of electrons, according to Algorithm 5.
 - (iii) The multipole information from all boxes in the interaction list is accessed by sorting the electrons in multiple Morton orderings, as illustrated in Fig. 3.

The order needed for the multipole method is $\mathcal{P} = \mathcal{O}(\log(1/\epsilon))$, which results in the complexity for the quantum FMM for each step of the product formula being (see Appendix A and Appendix B 4)

$$\mathcal{O}(\eta \log N \log^4(1/\epsilon)). \quad (8)$$

As this is linear in η , it is an asymptotic speedup over direct summation. The qubit usage scales as

$$\mathcal{O}(\eta \log N \log^3(1/\epsilon)) \quad (9)$$

and is primarily due to tree storage for the FMM.

for some kernel function Φ , where $V_i = \frac{1}{2} \sum_{j \in [\eta] \setminus \{i\}} \times \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j$. 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,

$$V = \sum_{i \in [\eta]} q_i V_i = \sum_{i \in [\eta]} q_i (V_{i,\text{near}} + V_{i,\text{far}}(\epsilon)). \quad (11)$$

Here, $V_{i,\text{near}}$ is evaluated exactly via summation over a number of geometrically close particles. The number of particles that are close to any given particle is bounded above by a constant. In contrast, $V_{i,\text{far}}$ 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 $\mathcal{O}(\eta \text{polylog}(1/\epsilon))$ 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.

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 organized so that the root node (1st level) encompasses the entire simulation cell, the leaf nodes (L th 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 ℓ th 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 (without loss of generality, those indexed in the sum over j in Eq. (10)) 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 (without loss of generality, those indexed in the sum over i in Eq. (10)) and

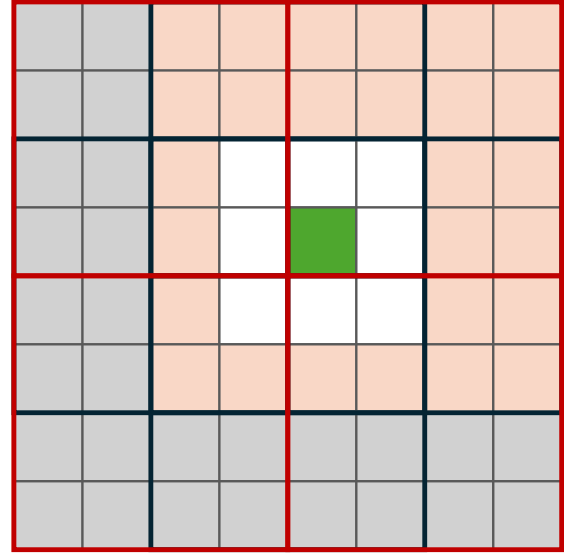


FIG. 1. Illustration of hierarchical division of a simulation cell, as well as the interaction list and neighbors for an exemplary (green) box in the $\ell = 4$ th level of a tree. Here, the level-1 box is the complete region. The four level-2 boxes are outlined by the red lines. The 16 level-3 boxes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

- (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 neighbors but their parent boxes (at the next level up) are. This is illustrated in Fig. 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 neighbors to b as $NN(b)$. Here, we define the nearest neighbors of b to be all boxes that share an edge or vertex with b . For $\ell \neq 1$, the

TABLE II. Summary of notation for boxes.

B_ℓ	set of boxes on level ℓ
$NN(b)$	nearest neighbors 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 center of box b

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

$$I(b) = C(NN(P(b))) - NN(b) - b, \quad (12)$$

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 center of box b . The notation introduced here is summarized 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 center 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 provide pseudocode that implements a simplified FMM that returns an approximation to V . This version of the algorithm only calculates the total far-field contribution to the potential between boxes and does not disaggregate those interactions to the local potential experienced by each particle. Later, we present a version of the algorithm that calculates these interactions via disaggregation (vide infra Algorithm 2). The algorithm consists of two passes through the hierarchical tree in which every box at levels $\ell \geq 3$ is iterated through. In the upward pass, the objective is to compute the charge Q_b for each box so that it can be used in the downward pass to compute an approximation of V . The upward 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

ALGORITHM 1. Classical FMM (monopole only, no disaggregation).

Upward pass: compute charges

1. **for** $b \in B_L$
 $Q_b \leftarrow \sum_{j \in p(b)} q_j$
end
2. **for** $\ell \leftarrow L - 1$ **to** 3
for $b \in B_\ell$
 $Q_b \leftarrow \sum_{a \in C(b)} Q_a$
end
end

Downward pass: evaluate potential
initialize $V \leftarrow 0$
Compute far-field contribution using box-box potentials

3. **for** $\ell \leftarrow 3$ **to** L
for $b \in B_\ell$
for $a \in I(b), a > b$
 $V \leftarrow V + Q_b \Phi(\mathbf{c}_a, \mathbf{c}_b) Q_a$
end
end
end

Compute near-field contribution using particle-particle potentials

4. **for** $b \in B_L$
for $i \in p(b)$
for $n \in NN(b), n > b$
for $j \in p(n)$
Compute Coulomb interactions between particles within b and n
 $V \leftarrow V + q_i \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j$
end
end
end
end
end

end
Return V

$\ell = L - 1$, 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 downward pass is to compute an approximation to V using the monopoles computed in the upward 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 upward pass. For all other levels, the interactions between a given box b and the

ALGORITHM 2. Classical FMM (monopole only, with disaggregation).

Upward pass: compute charges

1. **for** $b \in B_L$
 $Q_b \leftarrow \sum_{j \in p(b)} q_j$
end
2. **for** $\ell \leftarrow L - 1$ **to** 3
for $b \in B_\ell$
 $Q_b \leftarrow \sum_{a \in C(b)} Q_a$
end
end

Downward pass: evaluate potential

initialize $V_b \leftarrow 0$

Compute far-field contribution

3. **for** $\ell \leftarrow 3$ **to** L
for $b \in B_\ell$
 $V_b \leftarrow V_{P(b)}$
for $a \in I(b), a > b$
 $V_b \leftarrow V_b + \Phi(\mathbf{c}_a, \mathbf{c}_b) Q_a$
end
end

Compute near-field contribution

initialize $V \leftarrow 0$

4. **for** $b \in B_L$
for $i \in p(b)$
 $V_i \leftarrow V_b$
for $n \in NN(b), n > b$
for $j \in p(n)$
Compute Coulomb interactions between particles within b and n
 $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j$
end
end
for $j \in p(b), j > i$
Compute Coulomb interactions between particles within b
 $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j$
end
 $V \leftarrow V + q_i V_i$
end
end

Return V

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-neighbor boxes are computed exactly. All the interactions computed throughout the downward pass are aggregated in a variable V that is returned upon the algorithm completion.

In FMM, rather than simply summing the potential energy into V , the downward pass translates the multipole expansions into local expansions and disaggregates those local expansions into potentials at each leaf-level box.

In Algorithm 2 we present a version of the FMM using this disaggregation, though still using the monopole approximation for simplicity. It uses, for example, $V_b \leftarrow V_b + \Phi(\mathbf{c}_a, \mathbf{c}_b) Q_a$ rather than $V \leftarrow V + Q_b \Phi(\mathbf{c}_a, \mathbf{c}_b) Q_a$, with the multiplication by the box charge Q_b omitted to give the box potential rather than the potential energy. The potentials for parent boxes are copied to child boxes and eventually to the potentials for particles. These are then multiplied by the particle charges to determine the potential energy.

In the monopole-only case described here, these two approaches are completely equivalent, but in the higher-order calculation, the disaggregation simplifies the calculation. Determining the potential energy between boxes based on the multipole (as in the direct translation of Algorithm 1) would introduce significant additional complexity. The complexity is reduced by using the multipole information for the boxes in the interaction list to determine the potential at box b . A detailed explanation of the higher-order method is given in Appendix A.

Because each interaction list contains $\mathcal{O}(1)$ boxes, and each level- L box has $\mathcal{O}(1)$ neighbors (each with $\mathcal{O}(1)$ particles), the downward 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 to 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 Sec. V, we modify our approach to allow for adaptive grids, which lets us treat 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 $\tilde{\mathcal{O}}(\eta)$. For each box b , we have the registers:

- (i) $|Q_b\rangle$ —the total charge (monopole) of b ,
- (ii) $|V_b\rangle$ —the potential at box b .

In the higher-order FMM, Q_b and V_b will be replaced by the multipole and local expansion coefficients, respectively. In the monopole case, registers for V_b are not strictly needed because we could perform the algorithm in the simplified form in Algorithm 1. Quantum registers for V_i (the potentials at each particle) are not needed, because any particle's far-field contribution to the potential energy can be directly added into V without storing each V_i .

In addition to these registers, boxes at the leaf level have the additional registers:

- (i) $|\mathbf{r}_i\rangle, i \in p(b)-c$ registers storing the position of each particle,
- (ii) $|q_i\rangle, i \in p(b)-c$ 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 $k \in \{1, 2, \dots, L, \dots, n_p\}$, 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. 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 the 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 are in the correct boxes at level ℓ , then within each box the electron data are 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 generalize 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 initialized to $|1\rangle$. The remaining $n_b c - \eta$ particle registers are initialized to the all-zero state, so $|\mathbf{0}\rangle = |0 \dots 0\rangle$ 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 are 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

ALGORITHM 3. Sorting electron registers.

```

for  $l \leftarrow 0$  to  $3(L-1) - 1$ 
  for  $b \leftarrow 0$  to  $2^l - 1$ 
     $w \leftarrow c \times 2^{3(L-1)-l}$ 
     $j_0 \leftarrow bw$ 
    Sort registers  $j_0$  to  $j_0 + w - 1$ .
    for  $j \leftarrow j_0$  to  $j_0 + w - 1$ 
      if  $p_j \geq (j_0 + w/2)/c \times N/2^{3(L-1)}$ 
        Swap register  $j$  with register  $j + w/2$ .
      end
    end
  end
end

```

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 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 4 (which sums the potential over electrons in neighboring 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 $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{r}_j)q_j$, we need not directly store V_i and instead add $q_i\Phi(\mathbf{r}_i, \mathbf{r}_j)q_j$ into V . This means that we perform a Toffoli on the flag registers for electrons i and j and use the result to control the 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 $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{r}_j)q_j$ in loop 4, summing over nuclear charges as well. That is, q_j also includes the nuclear charges, and we can use q_i to control adding $\Phi(\mathbf{r}_i, \mathbf{r}_j)q_j$ to the total potential. 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 2) and summing contributions to the potential from boxes in the interaction list (loop 3). These operations all involve additions, except loop 3, which also requires a multiplication. The value of $\Phi(\mathbf{c}_a, \mathbf{c}_b)$ used in loop 3 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 to be calculated via coherent arithmetic is for the interparticle potentials in loop 4.

Overall the complexity of these calculations is as follows.

- (i) The complexity of adding the numbers of electrons in the n_b boxes is $\mathcal{O}(n_b c)$.
- (ii) 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 the total complexity $\mathcal{O}(n_b \log c)$.
- (iii) 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 the total complexity $\mathcal{O}(n_b \log c \log(1/\epsilon))$.
- (iv) 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 [38]. 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

$$\mathcal{O}(n_b c^2 \log^2(1/\epsilon) \log \log(1/\epsilon)). \quad (14)$$

- (v) Multiplying V_i by q_i for each electron and adding into V corresponds to a controlled addition, so it has total complexity $\mathcal{O}(\eta \log(1/\epsilon))$.

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

$$\mathcal{O}(\eta \log^2(1/\epsilon) \log \log(1/\epsilon)). \quad (15)$$

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. In addition, V_b is replaced with a number of local expansion coefficients scaling as the square of the order, and there is additional arithmetic needed to translate this information to child boxes. These features result 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 of a very small size, but no more than η of these boxes need to 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 of 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 generalize it to multiple dimensions. We also describe the method for the electron-electron potential V first, then an 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 are copied to a neighboring 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 neighboring 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 4. This is the quantum implementation of the upward pass of the FMM in Algorithm 2.

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 neighboring 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

ALGORITHM 4. Calculating charge information.

```

1. for  $j \leftarrow 0$  to  $\eta - 1$ 
    $Q(j, L) \leftarrow 1$  (corresponding to a single charge for this
   finest-level box)
end
2. for  $\ell \leftarrow L - 1$  to 3 by  $-1$ 
   (a) for  $j \leftarrow 0$  to  $\eta - 2$ 
       if  $f_b(j + 1, \ell) = f_b(j, \ell)$ 
         if  $f_b(j + 1, \ell + 1) \neq f_b(j, \ell + 1)$ 
            $Q(j + 1, \ell) \leftarrow Q(j, \ell + 1)$ 
         else if  $f_b(j, \ell + 1) \bmod 2 = 1$ 
            $Q(j + 1, \ell) \leftarrow Q(j, \ell)$ 
         end
       end
   end
   (b) for  $j \leftarrow \eta - 2$  to 0 by  $-1$ 
       if  $f_b(j, \ell) = f_b(j + 1, \ell)$ 
         if  $f_b(j, \ell + 1) \neq f_b(j + 1, \ell + 1)$ 
            $Q(j, \ell) \leftarrow Q(j + 1, \ell + 1)$ 
         else if  $f_b(j, \ell + 1) \bmod 2 = 0$ 
            $Q(j, \ell) \leftarrow Q(j + 1, \ell)$ 
         end
       end
   end
   (c) for  $j \leftarrow 0$  to  $\eta - 1$ 
        $Q(j, \ell) \leftarrow Q(j, \ell) + Q(j, \ell + 1)$ 
   end
end

```

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 that the operation in 2(c) to combine multipole information is not just addition.

The same principle can be used in higher dimensions. In 1D 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, even though we would be aiming to divide into four 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 neighboring regions rather than eight, but three levels give the addition of the charges from eight subcubes. Algorithm 4 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).

For the multipole information there is a translation used as well when aggregating the information for the boxes to the next level up. In 2D or 3D this translation will be to the center of the parent box, even though we are adding together information for each direction separately. This means that the translation operation need only be performed once for each change in level, similar to the classical FMM.

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 labeling 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 neighboring 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 are 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 neighbors, 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 neighboring 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 neighboring boxes at that level. Nevertheless, we can still use a similar approach for copying information from neighboring 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 neighboring 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 this procedure, registers are associated with each electron to store the charges of the boxes to the left of the electron's box. This information is then used to compute 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 neighbors is as shown in Algorithm 5.

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) is 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 neighbors 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 neighbor 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 contributions to the total potential energy that have already been accounted for, so it is not needed.

Algorithm 5 can also be used to calculate the charge/multipole information in 3D, instead of using the intermediate levels in the simplified method described above. That is, replace steps 2(a) and 2(b) in Algorithm 4 with Algorithm 5 (with $K = 7$) to copy information from registers that are in the same box at level ℓ but different at level $\ell + 1$. This will retrieve information for the eight boxes at level $\ell + 1$ necessary to calculate the charge/multipole information for each box at level ℓ .

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 Fig. 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 forward and backward 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 backward would only yield contributions to the potential that have already been accounted for when running through the list forward, 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. [40,41] use multiple Hilbert orderings, and Ref. [41] 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

ALGORITHM 5. Copying information for K -neighbors.

```

for  $j \leftarrow 0$  to  $\eta - 2$ 
  if  $f_b(j+1, \ell) = f_b(j, \ell)$ 
    for  $k \leftarrow 1$  to  $K$ 
       $Q_k(j+1, \ell) \leftarrow Q_k(j, \ell)$ 
    end
  else if  $f_b(j+1, \ell) \leq f_b(j, \ell) + K$ 
     $\Delta k \leftarrow f_b(j+1, \ell) - f_b(j, \ell)$ 
    for  $k \leftarrow \Delta k$  to  $K$ 
       $Q_k(j+1, \ell) \leftarrow Q_{k-\Delta k}(j, \ell)$ 
    end
  end
end

```

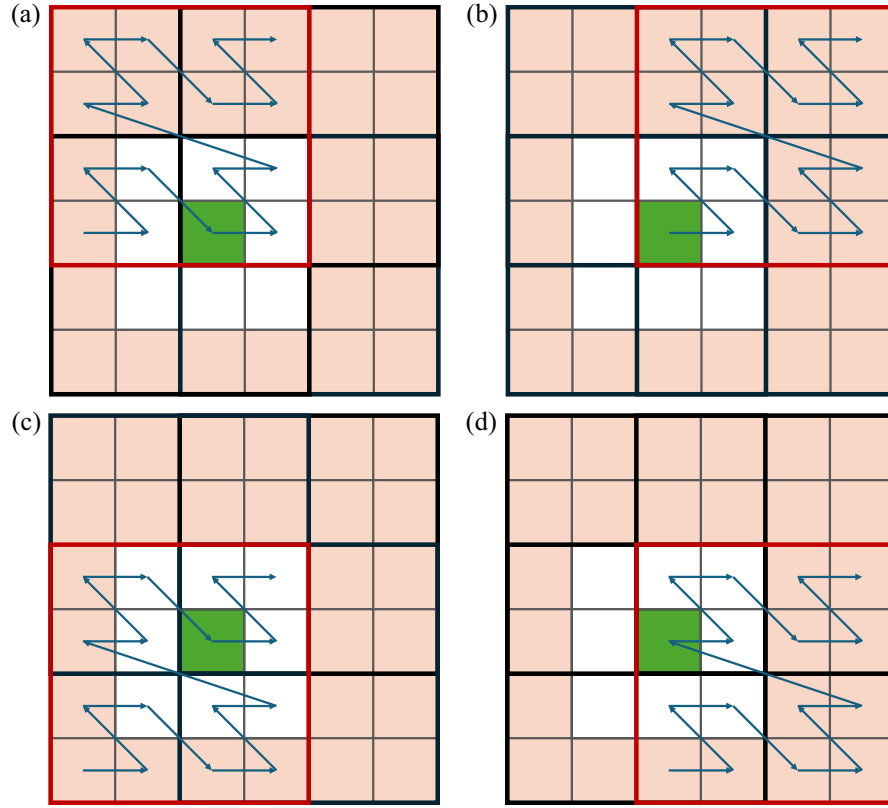


FIG. 3. The interaction list and neighbors used for FMM. The green square is the box b of interest, the white squares are neighbors of b , and the boxes in the interaction list are shown in orange. The heavy black lines indicate the boxes for the next level up. The red lines indicate the box two levels 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.

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 Fig. 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 Fig. 3(b). For example, if $\ell - 2 = 2$ and the first 3 bits of the x coordinates are

$$000, \quad 001, \quad 010, \quad 011, \quad 100, \quad 101, \quad (16)$$

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

$$010, \quad 011, \quad 100, \quad 101, \quad 110, \quad 111, \quad (17)$$

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 four 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 Fig. 3(b), then the increment in the x coordinate will shift the effective box to that shown in Fig. 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 $4^d - 1$ for dimension d 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

$$D_p := \{\mathbf{q} \in [0, 2^n - 1]^d \mid |p_j/2 - \lfloor q_j/2 \rfloor| \leq 1, \quad \forall j \in [1, d]\}, \quad (18)$$

and $M: \mathbb{N}^d \rightarrow \mathbb{N}$ 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

$$|M(\mathbf{p} + \mathbf{z} \bmod 2^n) - M(\mathbf{q} + \mathbf{z} \bmod 2^n)| \leq 4^d - 1. \quad (19)$$

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

$$\mathbf{z} = 2\lfloor \mathbf{p}/4 \rfloor - \lfloor \mathbf{q}/4 \rfloor. \quad (20)$$

Since $|\lfloor p_j/2 \rfloor - \lfloor q_j/2 \rfloor| \leq 1$, it must also be true that $|\lfloor p_j/4 \rfloor - \lfloor q_j/4 \rfloor| \leq 1$, 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

$$\begin{aligned} \lfloor (p_j + z_j \bmod 2^n)/4 \rfloor &= \lfloor p_j/4 \rfloor = \lfloor q_j/4 \rfloor \\ &= \lfloor (q_j + z_j \bmod 2^n)/4 \rfloor. \end{aligned} \quad (21)$$

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 $|\lfloor p_j/2 \rfloor - \lfloor q_j/2 \rfloor| \leq 1$ for the requirement that $\mathbf{q} \in D_p$. Then

$$p_j + z_j = 4a + r + 2 = 4(a + 1) + r - 2 = 4b + r - 2, \quad (22)$$

$$q_j + z_j = 4b + s + 2. \quad (23)$$

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

$$\begin{aligned} \lfloor (p_j + z_j \bmod 2^n)/4 \rfloor &= \lfloor (p_j + z_j)/4 \rfloor \\ &= \lfloor (4b + r - 2)/4 \rfloor = b \\ &= \lfloor (4b + s + 2)/4 \rfloor \\ &= \lfloor (q_j + z_j)/4 \rfloor \\ &= \lfloor (q_j + z_j \bmod 2^n)/4 \rfloor. \end{aligned} \quad (24)$$

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

$$\lfloor (p_j + z_j \bmod 2^n)/4 \rfloor = \lfloor (q_j + z_j \bmod 2^n)/4 \rfloor. \quad (25)$$

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 neighbors 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. (19) 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 6, which is the quantum implementation of loop 3 of the FMM algorithm in Algorithm 2. 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 neighboring boxes at level ℓ .

ALGORITHM 6. Calculation of potential in dimension d .

```

K ← 4d - 1
V ← 0
for ℓ ← 3 to L
  for  $\mathbf{z} \in \{0, 2\}^d$ 
    Add  $\mathbf{z}$  to box number at level  $\ell$  and sort according to
    Morton ordering.
    Apply Algorithm 5 to copy charge information.
    for j ← 0 to η - 1
      Vb ← VP(b)
      for k ← 1 to K
        if Qk(j, ℓ) contains charge information for a box in
        the interaction list not already added
          Vb ← Vb + Φ(ca, cb)Qk(j, ℓ)
        end
      end
    end
    Invert Algorithm 5 to erase Qk(j, ℓ) registers. Invert
    Morton ordering sort and subtract  $\mathbf{z}$ .
  end
end

```

As discussed above, we need only go through the list in one direction.

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 6, the quantity $\mathbf{c}_b$ is the position of the center of the level- ℓ box that particle j is in, and $\mathbf{c}_a$ is the center of the box that the information in $Q_k(j, \ell)$ corresponds to. The box potential V_b is stored in a register associated with each particle as well. Because registers associated with each particle contain the information for boxes it is contained in at each level, the information from the parent box may be easily copied. That is, we do not need to locate information in registers that are associated with other particles for $V_b \leftarrow V_{P(b)}$.

The remaining part needed is the direct calculation of the sum over pairwise interactions with particles in neighboring 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 neighboring boxes at level L is easily accounted for by including neighboring boxes in the interaction list in Algorithm 6. The final sum over $q_i V_i$ to obtain the potential energy V is easily obtained by summing over the values of V_b that have been calculated at level $\ell = L$. In this way, this algorithm effectively gives the quantum implementation of the downward pass of the FMM in Algorithm 2, with the quantum implementation of the upward pass of Algorithm 1 being given in Algorithm 4.

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 analyze 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 analyze the complexity for

calculating the electron-electron potential, because the nuclei do not change the complexity. In Algorithm 4 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

$$\mathcal{O}(\eta \log \eta \log N), \quad (26)$$

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. (26).

Next we consider the complexity of Algorithm 6 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

$$\mathcal{O}(\eta \log \eta \log^2 N). \quad (27)$$

2. We need to use Algorithm 5 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. (26).
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 $Q_k(j, \ell)$ 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. As a result, the total complexity of this part is

$$\mathcal{O}(\eta \log N [\log \eta \log(1/\epsilon) + \log N]). \quad (28)$$

In practice, the cost of multiplication is significantly larger than the cost of checking whether the box is in the interaction list, so the $\log N$ in the square brackets may be omitted.

4. The final sum over $q_i V_i$ is just a sum over the local potentials and it has complexity

$$\mathcal{O}(\eta \log(1/\epsilon)). \quad (29)$$

The main contributions to the complexity are from Eqs. (28) and (27). In this costing, the number of bits for the arithmetic is taken to be $\mathcal{O}(\log(1/\epsilon))$, so there is no need to take the number of bits for each component of the coordinates to be more than $\mathcal{O}(\log(1/\epsilon))$. This corresponds to a choice of N such that $\log N = \mathcal{O}(\log(1/\epsilon))$. Note that it is usually found that the error due to the finite size of basis sets scales as $\mathcal{O}(1/N)$ [42–48], so it is sufficient to take $N = \mathcal{O}(1/\epsilon)$. That also implies $\log N = \mathcal{O}(\log(1/\epsilon))$. In that case the complexity of the whole procedure can be given as $\mathcal{O}(\eta \log \eta \log N \log(1/\epsilon))$.

The leading contributions to the qubits used are as follows:

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

$$\mathcal{O}(\eta \log N). \quad (30)$$

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

$$\mathcal{O}(\eta \log \eta \log N). \quad (31)$$

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

3. The potentials V_b are stored for each level with $\mathcal{O}(\log(1/\epsilon))$ qubits each, for a total

$$\mathcal{O}(\eta \log N \log(1/\epsilon)). \quad (32)$$

4. 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

$$\mathcal{O}(\eta \log \eta). \quad (33)$$

The leading contribution to the logical qubits used is $\mathcal{O}(\eta \log N \log(1/\epsilon))$ to store the potentials V_b .

More generally, for the multipole expansion, the order needed for precision ϵ is $\mathcal{P} = \mathcal{O}(\log(1/\epsilon))$. 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 $\mathcal{O}(\mathcal{P}^2 \log(1/\epsilon)) = \mathcal{O}(\log^3(1/\epsilon))$. Similarly, V_b is replaced with a local expansion needing a number of bits $\mathcal{O}(\log^3(1/\epsilon))$

(see Appendix A). The total number of qubits used can therefore be given as

$$\mathcal{O}(\eta \log N \log^3(1/\epsilon)). \quad (34)$$

Similarly, the operations on the multipole information involve $\mathcal{O}(\mathcal{P}^2)$ arithmetic operations, which are primarily multiplications, so they have a complexity of $\mathcal{O}(\log^2(1/\epsilon))$. These operations are performed for each electron at $\log N$ levels, giving the complexity of the quantum FMM as

$$\mathcal{O}(\eta \log N \log^4(1/\epsilon)). \quad (35)$$

This complexity is true both for the calculation of the multipole information by combining that for individual boxes and the calculation of the potential (see Appendix B 4).

For the overall complexity, we need to account for the number of Trotter steps as well. This is accounted for in Ref. [18], which we summarize here. A real-space grid Hamiltonian may be given as

$$H = \sum_{j,k,\sigma} \tau_{j,k} a_{j,\sigma}^\dagger a_{k,\sigma} + \sum_{l,m,\sigma,\tau} \nu_{l,m} a_{l,\sigma}^\dagger a_{l,\sigma} a_{m,\tau}^\dagger a_{m,\tau}, \quad (36)$$

where $a_{j,\sigma}^\dagger$ 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 [18]

$$\begin{aligned} \|S_k(t) - e^{-itH}\|_{W_\eta} \\ = \mathcal{O}((\|\nu\|_{1,[\eta]} + \|\tau\|_1)^{k-1} \|\tau\|_1 \|\nu\|_{1,[\eta]} \eta t^{k+1}), \end{aligned} \quad (37)$$

where the norms are defined as

$$\|\nu\|_{1,[\eta]} = \max_j \max_{k_1 < \dots < k_\eta} (|v_{j,k_1}| + \dots + |v_{j,k_\eta}|), \quad (38)$$

$$\|\tau\|_1 = \max_j \sum_k |\tau_{j,k}|. \quad (39)$$

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

$$\mathcal{O}(t^{1+1/k} (\|\nu\|_{1,[\eta]} + \|\tau\|_1)^{1-1/k} (\|\tau\|_1 \|\nu\|_{1,[\eta]} \eta / \epsilon)^{1/k}). \quad (40)$$

The scaling of the norms is

$$\|\nu\|_{1,[\eta]} = \mathcal{O}\left(\frac{\eta^{2/3} N^{1/3}}{\Omega^{1/3}}\right), \quad (41)$$

$$\|\tau\|_1 = \mathcal{O}\left(\frac{N^{2/3}}{\Omega^{2/3}}\right). \quad (42)$$

This then gives the number of steps as

$$\mathcal{O}\left(t^{1+1/k}\left(\frac{\eta^{2/3}N^{1/3}}{\Omega^{1/3}} + \frac{N^{2/3}}{\Omega^{2/3}}\right)^{1-1/k}\left(\frac{\eta^{5/3}N}{\Omega\epsilon}\right)^{1/k}\right). \quad (43)$$

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

$$t\left(\frac{\eta^{2/3}N^{1/3}}{\Omega^{1/3}} + \frac{N^{2/3}}{\Omega^{2/3}}\right)\left(\frac{\eta N t}{\Omega\epsilon}\right)^{o(1)}. \quad (44)$$

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 $\text{polylog}(1/\epsilon)$ 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

$$t\left(\frac{\eta^{5/3}N^{1/3}}{\Omega^{1/3}} + \frac{\eta N^{2/3}}{\Omega^{2/3}}\right)\left(\frac{\eta N t}{\Omega\epsilon}\right)^{o(1)}, \quad (45)$$

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

$$t(\eta^{4/3}N^{1/3} + \eta^{1/3}N^{2/3})\left(\frac{\eta N t}{\epsilon}\right)^{o(1)}, \quad (46)$$

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. (43), 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 the simulation of quantum chemistry based on product formulas. The fast multipole method provides a speedup to $\tilde{\mathcal{O}}(\eta)$ 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 neighboring 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 [34]. That method, however, has an additional factor of η^3 , and so the method presented here has the best complexity for $N < \eta^7$, which is the typical range considered for realistic examples. For more comparable methods, such as the first-quantized product formula approach in Ref. [5], we improve over the complexity by a factor of η (up to logarithmic factors).

For these results on the FMM it is important that we are considering the first-quantized representation. For second quantization the double sum may be avoided in a much simpler way using the approach of Ref. [15], so there is no advantage to using the FMM. We provide better performance than the second quantization approach, because second quantization yields a complexity at least linear in N for the sum, which is much larger for the typical scenario where $N \gg \eta$.

In this work we have focused on showing that it is possible to achieve the asymptotic scaling associated with the FMM. In practice, there are large constant factors that mean thousands of electrons are needed for FMM to provide a speedup. There are further overheads involved in the quantum implementation, which could increase the regime for a speedup by an order of magnitude. Moreover, the implementation uses a large amount of data, around 160 million logical qubits for the example of $\eta = 4000$ electrons. That is for the memory-efficient form in Appendix B 4, where most of the storage is for multipole information. In comparison, second-quantized representations would require larger numbers of qubits for large N , since a qubit is needed for every orbital. Moreover, the method in Ref. [15] has a further space overhead of a factor of $\log(1/\epsilon)$ for the method of calculating the potential.

Therefore, any physical implementation will require access to a quantum computer with a number of physical qubits exceeding $\sim 10^8$. The scale of this overhead depends on assumptions about the hardware platform as well as the error correction scheme used. For example, an implementation in the surface code would require thousands of physical qubits per logical qubit, but this can be improved upon by using high-rate LDPC codes, such as lifted-product (LP) codes, which are far more compact [49]. For example, LP memory codes can have rates as high

as 0.285 [50], implying ~ 560 million physical qubits for data storage, with additional qubits needed for fault-tolerant execution of logical gates. That is large, though comparable to early estimates for simulation of FeMoco [51]. Similarly, further improvements of the implementation here could significantly reduce the qubit requirements, positioning the FMM as a viable algorithm for later generations of fault-tolerant devices. The coherent sorts used for data retrieval move data between a wide range of locations. That means a physical implementation would need more than nearest-neighbor connectivity to avoid data-movement overheads, though that would already be needed for LP codes.

However, the focus in this work is on showing that it is possible to achieve the $\mathcal{O}(\eta)$ complexity at all, and reducing the overheads is a topic for future work. In particular, we have used an early form of the FMM for clarity of exposition [32,53], and there have been many advances since then. For example, the Tucker decomposition [53] and reduced-rank approximations [54] can reduce the data needed for the FMM. Recent work has shown a “skeleton” representation of the far field can simplify the implementation [25]. FMM code developed for GPUs, such as PKDGRAV3, is designed to minimize data movement as well as reduce memory usage [55].

There are also potential improvements that can be made to the quantum implementation to improve the efficiency. Note that 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. [16]). 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 [40,41] or different shifts in the Morton ordering [41] could provide better performance.

The overheads could also be significantly reduced using the simplified quantum FMM for evenly distributed particles from Sec. IV. The difficulty in using that scheme is that the quantum simulation includes basis states with unevenly distributed particles, which is why we have presented the full adaptive scheme as our main approach. The full adaptive scheme has larger overhead due to using more levels, associating multipole information with every electron, and the difficulty of accessing the information from the interaction list. If it were possible to bound the error while restricting it to a subspace of evenly distributed particles, then the simpler scheme of Sec. IV could be used.

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. An important special case is the simulation

of periodic quantum systems in a dual plane-wave basis [35]. That results in the kernel

$$V(\mathbf{r}_i, \mathbf{r}_j) = \frac{4\pi}{\Omega} \sum_{\nu} \frac{\cos(k_{\nu}(\mathbf{r}_i - \mathbf{r}_j))}{k_{\nu}^2}, \quad (47)$$

which is quite different from the standard Coulomb interaction in a real-space basis that we have analyzed here. However, adapting FMM to this would involve straightforward modification of the interaction lists and certain translation operators, as in classical implementations [56–58].

The fast multipole method is also related to the method of hierarchical matrices [59], which allows matrix-vector multiplication of general kernel matrices $\Phi(\|\mathbf{r}_i - \mathbf{r}_j\|)$ of dimension n in time $\mathcal{O}(n \text{ polylog}(n))$. 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. D. W. B. worked on this project under a sponsored research agreement with Google Quantum AI. D. W. B. is also supported by Australian Research Council Discovery Project DP220101602. A. D. B. was supported by the DOE Office of Fusion Energy Sciences “Foundations for quantum simulation of warm dense matter” project. E. C. E. and A. T. 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 U.S. Government retains, and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this article or allow others to do so for U.S. 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>.

DATA AVAILABILITY

There are no publicly available research data or software supporting this manuscript. Requests for further information or data should be sent to the authors.

APPENDIX A: ADDITIONAL FMM IMPLEMENTATION DETAILS

The main text primarily considers simplified versions of the FMM, Algorithm 1 and Algorithm 2 that use zeroth-order expansions. However, higher-order expansions are essential to achieve 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

$$\sum_{l=0}^{\infty} \sum_{m=-l}^l (L_{lm} \|\mathbf{r}\|^l + M_{lm} \|\mathbf{r}\|^{-l-1}) Y_{lm}(\theta, \phi), \quad (\text{A1})$$

where (r, θ, ϕ) are standard spherical coordinates and Y_{lm} are the spherical harmonics. Multipole expansions are ones for which the only non-zero coefficients are in M_{lm} , and local expansions are ones for which the only non-zero coefficients are in L_{lm} . 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. That is, $M_{00}(b)$ is equivalent to Q_b in the main text, and $L_{00}(b)$ is equivalent to V_b . Higher-order implementations require machinery for storing and manipulating higher moments of the charge distribution and potential.

Algorithm 7 provides the details of an order- $\mathcal{P}$ implementation of FMM [52]. As compared to Algorithms 1 and 2, the new elements 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.

ALGORITHM 7. Classical FMM (order $\mathcal{P}$).

```

Initialize  $V \leftarrow 0$ 
Upward pass
1. for  $b \in B_L$ 
   for  $l \leftarrow 0$  to  $\mathcal{P}$ 
     for  $m \leftarrow -l$  to  $l$ 
       Initialize  $M_{lm}(b) \leftarrow 0$ 
       for  $j \in p(b)$ 
          $M_{lm}(b) \leftarrow M_{lm}(b) + q_j \|\mathbf{r}_j - \mathbf{c}_b\|^l Y_l^{-m}(\theta_j, \phi_j)$ 
       end
     end
   end
2. for  $\ell \leftarrow L - 1$  to 3
   for  $b \in B_\ell$ 
     Initialize  $M_{lm}(b) \leftarrow 0$ 
     for  $b' \in C(b)$ 
        $M_{lm}(b) \leftarrow M_{lm}(b) + (\mathcal{T}_{MM}(b, b') \circ M(b'))_{lm}$ 
     end
   end
3. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $b' \in I(b)$ 
        $L_{lm}(b) \leftarrow L_{lm}(b) + (\mathcal{T}_{ML}(b, b') \circ M(b'))_{lm}$ 
     end
     if  $\ell \neq L$ 
       for  $b' \in C(b)$ 
          $L_{lm}(b') \leftarrow (\mathcal{T}_{LL}(b, b') \circ L(b))_{lm}$ 
       end
     end
   end
4. for  $b \in B_L$ 
   for  $i \in p(b)$ 
     Initialize  $V_i \leftarrow 0$ 
     for  $l \leftarrow 0$  to  $\mathcal{P}$ 
       for  $m \leftarrow -l$  to  $l$ 
          $V_i \leftarrow V_i + \|\mathbf{r}_i - \mathbf{c}_b\|^l Y_l^m(\theta_i, \phi_i) L_{lm}(b)$ 
       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
     for  $j \in p(b), i > j$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{r}_j) q_j$ 
     end
      $V \leftarrow V + q_i V_i$ 
   end
5. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
6. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
7. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
8. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
9. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
10. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
11. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
12. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
13. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
14. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
15. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
16. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
17. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
18. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
19. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
20. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
21. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
22. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
23. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
24. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
25. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
26. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
27. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
28. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
29. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
30. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
31. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
32. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
33. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
34. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
35. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
36. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
37. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
38. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
39. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
40. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
41. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
42. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
43. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
44. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
45. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
46. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
47. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
48. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
49. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
50. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
51. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
52. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
53. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
54. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
55. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
56. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
57. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
58. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
59. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
60. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
61. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
62. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
63. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
64. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
65. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
66. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
67. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
68. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
69. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
70. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
71. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
72. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
73. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
74. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
75. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
76. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
77. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
78. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
79. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
80. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
81. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
82. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
83. for  $\ell \leftarrow 3$  to  $L$ 
   for  $b \in B_\ell$ 
     for  $i \in p(b)$ 
        $V_i \leftarrow V_i + \Phi(\mathbf{r}_i, \mathbf{c}_b) q_b$ 
     end
   end
84. for  $\ell \leftarrow 3$  to  $L$ 
   for</
```

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. We indicate the action of one of the translation operators with a $\circ$, e.g., $(\mathcal{T}_{ML}(b, b') \circ M(b'))_{lm}$ translates a multipole expansion about box b' to a local expansion about box b . To remain agnostic of the implementation details particular to different approaches to applying the translation operators, the $\circ$ operation implies some unspecified contraction of the translation operator against its operand to yield an output of the same rank. Generically, the translation operators only depend on the structure of the tree and the form of the potential (Φ), so they do not require any quantum data access. They can be classically precomputed with $\mathcal{O}(\log N \text{polylog}(1/\epsilon))$ overhead in time and memory, which is also typical in purely classical FMM implementations.

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 (M_{00}) 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 multipole-to-multipole (MM) and local-to-local (LL) translation operators (see loops 2 and 3 of Algorithm 7, 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 multipole-to-local (ML) translations (see loop 3 of Algorithm 7) in any of the $L - 1$ levels in which those translations take place. An eightfold cubic symmetry might also be exploited in implementations in which the ML translation operators strictly depend on the distance between centroids (i.e., any translation can be mapped to an equivalent one in the first octant). We also note that L_{00} will be the only non-zero component of the local expansion at the leaf level, so an efficient implementation will only use elements of the LL translation operator that contribute to this component.

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 $\tilde{\mathcal{O}}(\mathcal{P}^4)$ floating-point operations per translation for dense translation operators. Early on in the history of FMM, it was realized that ML translation operators applied using fast Fourier transforms or in a basis in which they're diagonal would only require $\tilde{\mathcal{O}}(\mathcal{P}^2)$ floating-point operations [52] per ML translation. Achieving this scaling in practice required mitigating certain numerical instabilities at the cost of additional implementation complexity [32]. Alternatively, formulations based on Cartesian tensors would require $\tilde{\mathcal{O}}(\mathcal{P}^6)$ floating-point operations per

ML translation but could benefit from other design trade-offs (e.g., exact MM and LL translation could be used to reduce the effective size of the interaction list and the number of ML translations) [60]. It also warrants noting that certain formulations might require complex arithmetic at intermediate levels, even if the resulting potentials are real. To remain agnostic to these details in the complexity analysis that follows, we will assume that the MM and LL translations both require $\tilde{\mathcal{O}}(\mathcal{P}^\alpha)$ floating-point operations and the ML translations will require $\tilde{\mathcal{O}}(\mathcal{P}^\beta)$ floating-point operations, where α and β are positive integers with values determined by the precise details of the associated translation operator implementations. 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

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

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 $\mathcal{O}(\epsilon)$.

In the quantum implementation of the full classical FMM, there are a number of simplifications that can be made. First, in the upward pass at the leaf level, the expansion coefficients are determined using the spherical harmonics Y_l^m in Algorithm 7. In the quantum implementation of the adaptive algorithm, at the leaf level, the box size corresponds to the grid resolution. This means that the particle position is the center of the box, so $\mathbf{r}_j - \mathbf{c}_b = 0$, and therefore the only non-zero entries are for $l = m = 0$. At the leaf level, we may simply take $M_{00}(b) = q_j$ as in the monopole case.

Moreover, this simplification may be made when converting L_{lm} to V_i in loop 4. Because $\mathbf{r}_j - \mathbf{c}_b = 0$, the only non-zero contribution is for $l = m = 0$, and we may set $V_i = L_{00}(b)$. A further simplification is because in loop 4 we do not need the individual values of V_i , and so may add $q_i L_{00}(b)$ to V . Because of the encoding, $L_{00}(b)$ is only stored for charges $q_i = 1$, and so we simply add $L_{00}(b)$. Similarly, for nearest-neighbor boxes, we may simply add $\Phi(\mathbf{r}_i, \mathbf{r}_j)$. The data for these boxes are only present for particles, so the charge need not be explicitly multiplied. Then the last part in loop 4 is for other particles in the same leaf level box, which may simply be omitted in the quantum implementation.

As compared to the quantum implementation of the monopole algorithm, the key differences in the complexity are as follows:

- (1) The initial complexity in loop 1 of setting $M_{00}(b)$ equal to the charges is trivial.
- (2) When combining the multipole information from child boxes in loop 2, we have $\mathcal{O}(\log N)$ steps where a sort is performed with complexity $\mathcal{O}(\eta \log \eta)$ steps, but each step has its complexity increased. Rather than the items of data to be sorted having size $\mathcal{O}(\log N)$, we need to include $M_{lm}(b)$. That has size $\mathcal{O}(\mathcal{P}^2 \log(1/\epsilon))$, given that each entry of $M_{lm}(b)$ is given to $\mathcal{O}(\log(1/\epsilon))$ bits. Taking $\mathcal{P} = \mathcal{O}(\log(1/\epsilon))$, we then replace $\log N$ with $\log^3(1/\epsilon)$. Note that these sorts need only include information from $M_{lm}(b)$ for two levels (the parent and child), so we do not incur any extra factor of $\log N$ for the number of levels. The complexity of the sorts is therefore

$$\mathcal{O}(\eta \log \eta \log N \log^3(1/\epsilon)). \quad (\text{A3})$$

- (3) In order to combine information from child boxes in loop 2, we now need to perform a MM translation to determine the contribution to $M_{lm}(b)$ for the parent box. This requires $\tilde{\mathcal{O}}(\mathcal{P}^\alpha)$ operations, the cost of which will be dominated by multiplications which have complexity $\mathcal{O}(\log^2(1/\epsilon))$ in terms of elementary gates. There are $\mathcal{O}(\eta \log N)$ steps, and so the complexity is

$$\tilde{\mathcal{O}}(\eta \log N \log^{2+\alpha}(1/\epsilon)). \quad (\text{A4})$$

In this expression, $\tilde{\mathcal{O}}$ accounts for logarithmic factors in the MM translation complexity.

- (4) When performing the sorts to access the information from boxes in the interaction list, rather than just the box charge we need to include $M_{lm}(b)$ for the given level ℓ . The complexity of this sort is therefore equivalent to that used in loop 2 and is given by Eq. (A3).
- (5) When using Algorithm 5 to copy the multipole information, $\log \eta$ is replaced with $\log(1/\epsilon)$, and so the complexity becomes

$$\mathcal{O}(\eta \log N \log^3(1/\epsilon)). \quad (\text{A5})$$

- (6) Rather than just adding to the potential, we need the ML translation to give the contribution to $L_{lm}(b)$ from $M_{lm}(b')$ for the box in the interaction list. This requires $\tilde{\mathcal{O}}(\mathcal{P}^\beta)$ operations that will also be dominated by multiplications, and so Eq. (28) is replaced with

$$\tilde{\mathcal{O}}(\eta \log N \log^{2+\beta}(1/\epsilon)). \quad (\text{A6})$$

- (7) A task which had trivial complexity in the monopole version was copying the local data to the child boxes (because it was assumed the potential was unchanged). Here, that now requires LL translations, which has the same complexity as the MM translations above. Therefore, the complexity is again given by Eq. (A4).
- (8) The final additions into the total potential energy in loop 4 are essentially unchanged, because we are adding $L_{00}(b)$ as well as $\Phi(\mathbf{r}_i, \mathbf{r}_j)$, which is given by QROM. Therefore the complexity is

$$\mathcal{O}(\eta \log(1/\epsilon)). \quad (\text{A7})$$

The dominant contributions to the complexity are from the translations, which will have complexity

$$\begin{aligned} & \tilde{\mathcal{O}}(\eta \mathcal{P}^{\max(\alpha, \beta)} \log N \log^2(1/\epsilon)) \\ &= \tilde{\mathcal{O}}(\eta \log N \log^{2+\max(\alpha, \beta)}(1/\epsilon)). \end{aligned} \quad (\text{A8})$$

Here, the right expression is from taking the truncation level to be $\mathcal{O}(\log(1/\epsilon))$. There are also complexities with a factor of $\log \eta$ for sorts, but that complexity should be smaller because η should be smaller than $1/\epsilon$. It is needed to store data for M_{lm} and L_{lm} at each level and for each particle, and so the qubit usage is

$$\mathcal{O}(\eta \mathcal{P}^2 \log N \log(1/\epsilon)) = \mathcal{O}(\eta \log N \log^3(1/\epsilon)). \quad (\text{A9})$$

We give further details of these costings with example parameters in Appendix B.

We note another caveat in adapting existing classical implementations of FMM to a quantum setting. We consider one of the canonical FMM implementations described in Ref. [32]. There, the cost of tree traversal is dominated by the MM and LL translations for which $\alpha = 3$, whereas $\beta = 2$ for the ML translations. Nevertheless, the overall scaling for tree traversal is $\mathcal{O}(\mathcal{P}^2)$ (not $\mathcal{O}(\mathcal{P}^3)$) because the number of translations is taken to scale inversely with $\mathcal{P}$. This is a common trick in many implementations that is accomplished by letting the leaf-level particle density scale with $\mathcal{P}$, effectively shortening the tree. While it is still possible to shorten the tree by scaling the number of grid points per leaf box with $\mathcal{P}$ in our quantum implementation, this only impacts time and memory complexities by $\mathcal{O}(\log(\mathcal{P}))$ factors, i.e., substituting $N \rightarrow N/\mathcal{P}$ in Eqs. (A3)–(A9). Thus, a straightforward adaptation of the scheme in Ref. [32] will instead have $\mathcal{O}(\mathcal{P}^3)$ cost, and we expect that any classical implementation that achieves better asymptotics by adjusting the leaf-level density will suffer a similar drawback. Still, adapting the approach from Ref. [32] is asymptotically superior to the $\mathcal{O}(\mathcal{P}^4)$ cost that would be realized by a totally naive approach to translation. But borrowing

well-established tricks from the classical literature might not perform as well as expected.

Another compelling approach is to avoid ML and LL translations entirely, evaluating the far-field contribution to the potential at each particle directly from multipole expansions aggregated by MM translation alone. This is described in Appendix B 4. Regardless of the MM translation scheme used, the far-field potential evaluation from the multipole moments requires $\mathcal{O}(\eta \mathcal{P}^2 \log N \log^2(1/\epsilon))$ operations. However, this requires the use of coherent arithmetic to evaluate solid harmonics in lieu of using QROM to access precomputed ML and LL translation operators. Nevertheless, we will show that this appears to be superior to a straightforward quantum implementation of one of the canonical classical implementations [32].

APPENDIX B: BREAK-EVEN POINT FOR SPEEDUP OVER DIRECT SUMMATION

In this appendix, we estimate the constant factors required to determine the break-even point beyond which the FMM will provide a speedup over direct pairwise summation. This is heavily implementation dependent, so we will consider a quantum version of Greengard and Rokhlin's implementation in Ref. [32]. This requires cost models for the number of operations involved in direct pairwise summation, $\mathcal{C}_{\text{direct}}(\epsilon)\eta^2$, and the FMM implementation, $\mathcal{C}_{\text{tree}}(\epsilon)\eta$. Here, $\mathcal{C}_{\text{direct}}$ is the cost of evaluating the Coulomb potential for a single pair of particles and $\mathcal{C}_{\text{tree}}$ is the cost of tree traversal in the FMM. We are concerned with identifying the value of η beyond which $\mathcal{C}_{\text{direct}}(\epsilon)\eta^2 > \mathcal{C}_{\text{tree}}(\epsilon)\eta$ for a fixed value of ϵ . This is

$$\eta_{\text{BE}}(\epsilon) = \mathcal{C}_{\text{tree}}(\epsilon)/\mathcal{C}_{\text{direct}}(\epsilon). \quad (\text{B1})$$

The cost of evaluating the Coulomb potential for a single pair of particles is $\mathcal{O}(\log^2(1/\epsilon))$ [38]. The cost of tree traversal is primarily determined by the scaling with $\mathcal{P} \sim \mathcal{O}(\log(1/\epsilon))$, which is implementation dependent as discussed in Appendix A.

Data from Greengard and Rokhlin's classical implementation are consistent with $\mathcal{P} \approx 0.8 \log_2(1/\epsilon)$ [32]. While we should expect that an equivalent quantum implementation will make use of a taller tree, this should not dramatically change the constant factor coefficient to $\log_2(1/\epsilon)$ because L is also $\mathcal{O}(\log(1/\epsilon))$. However, we will see that this can have a significant impact on the cost of data movement in the quantum implementation. Greengard and Rokhlin also report the following data for $\eta_{\text{BE}}(\epsilon)$:

- (i) $\eta_{\text{BE}}(\epsilon = 4.5 \times 10^{-3}) = 500$, for $\mathcal{P} = 5$.
- (ii) $\eta_{\text{BE}}(\epsilon = 1.4 \times 10^{-4}) = 2000$, for $\mathcal{P} = 9$.
- (iii) $\eta_{\text{BE}}(\epsilon = 1.1 \times 10^{-7}) = 4000$, for $\mathcal{P} = 18$.
- (iv) $\eta_{\text{BE}}(\epsilon = 6.2 \times 10^{-12}) = 5000$, for $\mathcal{P} = 30$.

This does not necessarily translate directly to the break-even point for a quantum implementation because of the additional costs associated with retrieving data for the interaction list during tree traversal. Because the classical implementation can make use of fast access to a pre-computed octree during tree traversal, it is important to consider the relative cost of data movement and the coherent arithmetic involved in $\mathcal{C}_{\text{tree}}(\epsilon)$ before extrapolating from the classical implementation.

1. Data retrieval costs

There are additional costs involved in retrieving the data in the quantum implementation.

- (i) For 3D simulations, there are 8 sorts used to retrieve the data from the interaction list.
- (ii) For each of those sorts we run through the list of particles once, and account for particles that may be shifted by up to 63 boxes in 3D.

To compare the complexity for these operations to the complexity for the operations, we will consider the ratio of the gate complexity to the size of the data for the multipole information. Here we are considering only a single level at a time, and there is multipole information for each electron of size approximately $\mathcal{P}^2 \log_2(1/\epsilon)$.

For the sorts, the complexity is $\mathcal{O}(\eta \log \eta)$ comparators, and for small $\eta \leq 32$ highly optimized sorting networks are known. For larger η of thousands a systematic construction is needed, and the $\mathcal{O}(\eta \log \eta)$ constructions have very large constants. For example the Zig-zag Sort of Goodrich has a best-case complexity of $2700\eta \log_2 \eta$ [61]. In comparison, Batchers's odd-even mergesort [62] has complexity

$$\frac{1}{4}\eta \log_2^2 \eta - \frac{1}{4}\eta \log_2 \eta + \eta - 1, \quad (\text{B2})$$

giving complexity approximately 34η for $\eta = 4096$. The theoretical lower bound is $\lceil \log_2(\eta!) \rceil$, which gives about 10.6η for $\eta = 4096$, so improved sorting networks could reduce the complexity by at most a factor of 3.

For these sorts there are controlled swaps of the multipole information for each comparator. We also need to move the information for the local expansion, which is the same size as the multipole information. There is another factor of 2 in order to invert the sorts. As a result, the combined complexity/data ratio for the sorts is about $2 \times 2 \times 8 \times 34 = 1088$.

When running through the list of particles, the primary cost is copying data into appropriately shifted registers using Algorithm 5. These registers are used for multipole information from nearby boxes, so it is possible to retrieve information from the interaction list. In the following discussion, we will call these registers the partial interaction (PI) list as they only include information from part of the interaction list. As written in Algorithm 5, the value

of Δk is calculated, then data are copied using Δk to give the data location. The extra overhead of $K = 63$ for coherent data access can be avoided by using a cyclic shift, as for example given in Ref. [63]. That is, the bits of Δk are used to control a series of swaps to move the data by Δk positions.

The procedure can be further simplified by using a single PI list, rather than copying and shifting it for the next electron. The method is to include the position and multipole information for electron j with the PI list, compute Δk , then use it to control the cyclic shift of these 64 items, including the information for electron j . Items 2 to 64 are then used for the PI list for electron $j + 1$. If electron $j + 1$ is in the same box as j , then there is no shift, and the multipole information for electron j remains in that data location. If there is a shift, then the multipole information for electron j is replaced with information from the PI list that is not needed for electron $j + 1$.

In this way there is no cost of copying to a new list, and also only one copy of the PI list need be retained, rather than needing copies of the PI list for each electron. The values of Δk need be kept for each electron to invert the procedure, but only 6 qubits are needed for each value of Δk , making it a negligible qubit cost. Because the PI list is updated for each electron, the multipole-to-local translations to update the local expansion must be performed before proceeding to the next electron.

There is an overhead of 6 for the cost of the procedure shifting the entries in the PI list, corresponding to the number of bits of Δk . Therefore, for copying the neighbor information, there is a total complexity/data ratio of $2 \times 8 \times 6 \times 64 = 6144$, where the factor of 2 is for inverting the procedure, 8 is for the different sorts, 6 is for the controlled shift, and 64 is the size of the PI list being shifted.

So far we have just considered the movement of the multipole and local expansion. As will be discussed in the next subsection, the gate complexity is reduced by computing outgoing exponential expansions from the multipole information. The best gate complexity is obtained by calculating these for all boxes first, rather than calculating every time a box in the interaction list needs to be accessed. This means that the outgoing exponential expansions would need to be moved in the sort and in shifting the entries in the PI list, rather than the multipole information.

The size of the data for the exponential expansions is approximately $2 \times 6 \times P_{\text{exp}}(\epsilon) \log_2(1/\epsilon)$, where the factor of 2 is for complex numbers, 6 is for the 6 faces of a cube, and $P_{\text{exp}}(\epsilon) \approx 1.72\mathcal{P}^2$ is the number of basis functions. This results in the size of the data for the exponential expansion being about a factor of 18.5 times larger than the data for the multipole information (with $\mathcal{P} = 18$). This results in the complexity/data ratio of 1088 above for the sorts becoming 10,621, and the ratio of 6144 for the PI list becoming 113,815.

2. Coherent arithmetic costs

The costs of the arithmetic involved in a quantum implementation of the tree traversal scheme in Ref. [32] are described in what follows. Reference [32] considers several approaches to tree traversal: (1) one used in the original FMM [52] with straightforward translations, (2) a refinement in which the cost of translations is reduced through the use of rotation matrices, and (3) a further refinement that uses exponential expansions to reduce the cost even more. Approach (3) has the best scaling in $\mathcal{P}$ and it is the method that was used in generating the break-even points above, so it is what we consider here.

Computing the $\ell = L$ multipole expansions from the charges and the potential from the $\ell = L$ local expansions has negligible cost, because only the monopole moments contribute. All other MM and LL translations require $\mathcal{O}(\mathcal{P}^3)$ operations and the ML translations require $\mathcal{O}(\mathcal{P}^2)$ operations. The MM and LL translations are reduced from $\mathcal{O}(\mathcal{P}^4)$ operations due to the use of rotation matrices that orient the expansions such that each translation involves two sets of $\mathcal{O}(\mathcal{P}^2)$ multiplications and three sets of $\mathcal{O}(\mathcal{P}^3)$ multiplications. The ML translations are more complicated. They require converting each multipole expansion into an outgoing exponential expansion, translating in that exponential basis, and converting the resulting incoming exponential expansion into a local expansion. Each outgoing and incoming exponential expansion is itself decomposed into six distinct directions (one for each face of a box) and a quadrature rule with $\mathcal{O}(\mathcal{P}^2)$ points is used to evaluate an integral representation of the Coulomb potential, implementing translation in the exponential basis. It should be noted that the ML interaction list for any given box also needs to account for which of the six outgoing and six incoming expansions are properly oriented, i.e., an outgoing exponential expansion leaving one face will only be “received” by incoming exponential expansions on oppositely oriented faces. Overall, the multipole-to-exponential and exponential-to-local expansions each involve $\mathcal{O}(\mathcal{P}^3)$ multiplications, which only need to be performed once per box. Each translation in the exponential basis requires $\mathcal{O}(\mathcal{P}^2)$ multiplications and there are at most 189 such translations per box.

We proceed to count the number of multiplications *per particle* during tree traversal. It is impractical to account for all of the implementation optimisations that are described in Ref. [32], but we expect to be within an order of magnitude. For the purposes of estimating the overhead over a classical implementation, this should suffice. The number of levels where this full summation is performed is $L - 3$, so we give that factor in the following estimates. There is also a direct summation at the leaf level which will be considerably lower cost than the multipole calculations. Level $\ell = 3$ of the multipole calculation involves fewer boxes in the interaction list, but we omit that cost saving for simplicity.

- (1) Each MM translation requires two rotations about the z axis ($\mathcal{P}(\mathcal{P}+1)$ multiplications each), two rotations about the y axis ($((\mathcal{P}+1)(2\mathcal{P}+1)(2\mathcal{P}+3)/3$ multiplications each), and one translation along the z axis ($((\mathcal{P}+1)(\mathcal{P}+2)(4\mathcal{P}+3)/6$ multiplications). Thus, there are $(\frac{10}{3}\mathcal{P}^3 + \frac{25}{2}\mathcal{P}^2 + \frac{73}{6}\mathcal{P} + 3)(L-3)$ multiplications per particle due to MM translations up the tree.
- (2) Each real multipole expansion will be converted into six complex outgoing exponential expansions. Each exponential expansion is given in terms of $P_{\text{exp}}(\epsilon)$ basis functions, where $P_{\text{exp}}(\epsilon) \sim \mathcal{O}(\mathcal{P}^2)$ is the number of quadrature points used to discretise an integral representation of the Coulomb kernel. For accuracy consistent with $\mathcal{P} = 18$ (i.e., $\epsilon = 1.1 \times 10^{-7}$), $P_{\text{exp}}(\epsilon) = 558$ in Ref. [32], which implies $P_{\text{exp}}(\epsilon) \approx 1.72\mathcal{P}^2$. The number of multiplications involved in converting the multipole expansion to each of the six outgoing exponential expansions will vary, due to the need to apply different numbers of rotation matrices to align with different directions of propagation.
 - (a) There is one conversion that does not involve any rotations, and thus it requires $\mathcal{P}(\mathcal{P}+1)^2 + 2\mathcal{P}P_{\text{exp}}(\epsilon)$ multiplications. The factor of two in the second term accounts for the fact that each operation is actually a multiplication between a real number and a complex number. In this case and henceforth, the multiplications are ordered to minimize the number of complex multiplications.
 - (b) There are three conversions that also involve y -axis rotations, requiring $(\mathcal{P}+1)(2\mathcal{P}+1)(2\mathcal{P}+3)/3 + \mathcal{P}(\mathcal{P}+1)^2 + 2\mathcal{P}P_{\text{exp}}(\epsilon)$ multiplications each.
 - (c) There are two conversions that involve both y - and z -axis rotations, with $\mathcal{P}(\mathcal{P}+1) + (\mathcal{P}+1)(2\mathcal{P}+1)(2\mathcal{P}+3)/3 + \mathcal{P}(\mathcal{P}+1)^2 + 2\mathcal{P}P_{\text{exp}}(\epsilon)$ multiplications each.
 Thus, there are $(12P_{\text{exp}}(\epsilon)\mathcal{P} + \frac{38}{3}\mathcal{P}^3 + 34\mathcal{P}^2 + \frac{79}{3}\mathcal{P} + 5)(L-3)$ multiplications per particle for the multipole-to-exponential conversions.
- (3) Each outgoing-to-incoming ML translation requires $3P_{\text{exp}}(\epsilon)$ multiplications, where the factor of 3 reflects the fact that these are multiplications between complex numbers. There are at most 189 interacting pairs that require ML translation for any given box and 6 pairs of outgoing/incoming exponential expansions per box. Thus, there are at most $3402P_{\text{exp}}(\epsilon)(L-3)$ multiplications involved in the outgoing-to-incoming translations, across the tree.

- (4) Conversion of the six incoming exponential expansions mirrors the cost of the conversion of the multipole expansions into outgoing exponential expansions. Thus, there are $(12P_{\text{exp}}(\epsilon)\mathcal{P} + \frac{38}{3}\mathcal{P}^3 + 34\mathcal{P}^2 + \frac{79}{3}\mathcal{P} + 5)(L-3)$ multiplications per particle for the exponential to LL conversions. We note that the data become purely real again during this step.
- (5) Each LL translation has the same cost as an MM translation. Thus, there are $(\frac{10}{3}\mathcal{P}^3 + \frac{25}{2}\mathcal{P}^2 + \frac{73}{6}\mathcal{P} + 3)(L-3)$ multiplications per particle for the LL translations down the tree.

The total number of multiplications per particle for tree traversal then becomes

$$\begin{aligned} & ((3 \times 6 \times 189 + 24\mathcal{P})P_{\text{exp}}(\epsilon) + 32\mathcal{P}^3 + 93\mathcal{P}^2 + 77\mathcal{P} + 16) \\ & \times (L-3) \approx (73\mathcal{P}^3 + 5944\mathcal{P}^2 + \mathcal{O}(\mathcal{P}))(L-3). \end{aligned} \quad (\text{B3})$$

The quantum implementation is more costly than this, because rather than directly accessing all boxes in the interaction list, we are accessing the boxes through eight different sorts. In each of these sorts we retrieve 63 neighbors, though at most 56 may be in the interaction list. This means that the outgoing-to-incoming ML calculations in step 3 need to be performed 448 times, rather than 189. A further overhead is that the ML translation in step 3 should be uncomputed before the next step, to avoid needing to store these intermediate results of the calculation in ancillae. As a result, the 189 in Eq. (B3) should be replaced with 448×2 .

A further difference is that the incoming exponential expansions should be converted for each of the eight sorts. That is because the data for the incoming exponential expansions need to be uncomputed to erase the ancillae this information is stored in, and that uncomputation needs to be performed before the next sort. (Otherwise the incoming exponential expansions would need to be stored for every particle for every level, greatly increasing the data storage requirements.) As a result there is a factor of 16 on the complexity of step 4. We also uncompute the exponential expansion in step 2, giving a further factor of 2 for that step. Combining these considerations gives the complexity for the quantum tree traversal as

$$\begin{aligned} & \left((3 \times 6 \times 896 + 216\mathcal{P})P_{\text{exp}}(\epsilon) + \frac{704}{3}\mathcal{P}^3 + 637\mathcal{P}^2 + \frac{1495}{3}\mathcal{P} + 96 \right) \\ & \times (L-3) \approx (606\mathcal{P}^3 + 28377\mathcal{P}^2 + \mathcal{O}(\mathcal{P}))(L-3). \end{aligned} \quad (\text{B4})$$

Considering the ratio of the complexity to the size of the multipole data, there is an extra factor of $\log_2(1/\epsilon)$ for translating multiplications to elementary gates. There

would be an ratio of around $(73 \times 18 + 5944) \times 22 = 159,676$ over the size of the data for coherent arithmetic with 22 bits, were it possible to work directly with the interaction list. Accounting for the retrieval of data using sorts, as well as uncomputation costs, the ratio becomes $(606 \times 18 + 37624) \times 22 = 864,274$. Including the overhead for the sorts and data retrieval, this means that the coherent implementation has a factor of about 5.4 larger complexity than the classical implementation (as estimated in Eq. (B3)).

It is also possible to compute the outgoing exponential expansions (step 2) when building the PI list. That is, every time multipole information is added to the PI list, the outgoing exponential expansion is calculated and added to the PI list as well. Then when multipole information is removed from the PI list the corresponding outgoing exponential expansion is uncomputed. This greatly reduces the amount of data that need be stored, but the cost of step 2 is multiplied by 16 (due to the 8 sorts) instead of 2, giving the complexity for the quantum tree traversal as

$$\begin{aligned} & ((3 \times 6 \times 896 + 384\mathcal{P})P_{\text{exp}}(\epsilon) + 412\mathcal{P}^3 + 1113\mathcal{P}^2 + 867\mathcal{P} + 166) \\ & \times (L-3) \approx (1072\mathcal{P}^3 + 28853\mathcal{P}^2 + \mathcal{O}(\mathcal{P}))(L-3). \end{aligned} \quad (\text{B5})$$

This costing results in a complexity/data ratio of about 10^6 or about 6.6 times that in Eq. (B3).

3. Other additional costs

Another factor that may cause the quantum implementation to be more costly than that given in Ref. [32] is that it uses more levels. We consider a number of levels sufficient to resolve to the grid points, so for example 7 levels for 2^{21} grid points. In comparison, the number of levels given in Ref. [32] was about half that. There will also be an overhead because we have associated multipole information with each electron. In practice much of the advantage of the FMM is because many electrons are grouped together in each box, greatly reducing the number of boxes below the number of particles. Together with the factor of about 5 due to other quantum overheads from the above subsection, the threshold value of η where an improvement is obtained would be increased by at least an order of magnitude.

Another issue that should be considered in the practicality of the quantum FMM is the number of qubits used. To give a rough estimate, we will consider the example of order-18 FMM, 22 bits of precision, $\eta = 4000$ electrons, and $N = 2^{21}$. Given an FMM order of 18 and 22 bits of precision, there would be about 8000 qubits for the multipole information for each electron. With $N = 2^{21}$ there would be 5 levels to store multipole information for, giving a total of about 160 million qubits for the multipole expansion. There will also be a similar amount of

information for the local expansion, which would mean about 320 million qubits.

Note that we need only include storage costs for multipole information at each level, whereas Sec. VA presented a simplified scheme which involves calculation of charge/multipole information at intermediate levels. As noted in Sec. VB, one may use Algorithm 5 to retrieve information from all 8 child boxes to compute multipole information for a parent box, so intermediate levels are not needed. Similar to the PI list, a single list should be used, rather than a separate list for each electron.

In the above implementation, we first consider the case that the outgoing exponential expansions are stored at each level as well, but only one level at a time before being uncomputed. That would need about 150,000 qubits for each electron, making a total of 590 million qubits. The total qubit usage would then be about 910 million. Alternatively, we may just store the outgoing exponential expansions for the PI list, and uncompute them when they are removed from the list. That reduces the number of qubits for the outgoing exponential expansions to about 9 million, for a total of about 330 million qubits, though it would increase the gate cost.

4. Multipole-to-particle evaluation

Given the data usage and overheads involved with using the full FMM with ML and LL translations, it may be preferable to perform a direct multipole-to-particle calculation of the potential energy. Here we show that this approach significantly reduces both the qubit and gate cost. The method is to evaluate the contribution to the potential energy at particle i due to its interaction with particles in box a as

$$V_{i,a} = q_i \sum_{l=0}^{\mathcal{P}} \sum_{m=-l}^l M_{lm}(a) \|\mathbf{r}_i - \mathbf{c}_a\|^{-l-1} Y_{lm}(\theta_{i,a}, \phi_{i,a}), \quad (\text{B6})$$

without the use of a local expansion. This is particularly useful in the quantum implementation, because we perform the calculation for the multipole information associated with each electron. A major advantage of this approach is that it is no longer necessary to store the outgoing expansion information for all particles, which greatly increases the data storage requirements in the full method. Moreover, it is no longer necessary to store the local expansions. The majority of the data storage requirement is then just the multipole information, which reduces the qubit requirement to about 160 million for 4000 electrons. There is then the possibility for compressed representations to reduce the data requirement further [54,55].

It is relatively straightforward to evaluate each term in Eq. (B6). The irregular solid harmonics may be computed in real form using the following recurrence relations [64].

In the following we denote the real and imaginary components of $r^{-l-1}Y_{lm}$ by C_{lm} and S_{lm} respectively, with x, y, z being the components of the vector of length r .

- (1) The recursion starts with the monopole term for $l = 0, m = 0$:

$$C_{0,0} = \frac{1}{r}, \quad S_{0,0} = 0. \quad (\text{B7})$$

- (2) Then the terms along the diagonal for $m = l > 0$ are computed using

$$C_{l,l} = \frac{2l-1}{r^2} (xC_{l-1,l-1} - yS_{l-1,l-1}), \quad (\text{B8})$$

$$S_{l,l} = \frac{2l-1}{r^2} (xS_{l-1,l-1} + yC_{l-1,l-1}). \quad (\text{B9})$$

- (3) The terms in the off-diagonal for $m = l-1$ are given by

$$C_{l,l-1} = \frac{(2l-1)z}{r^2} C_{l-1,l-1}, \quad (\text{B10})$$

$$S_{l,l-1} = \frac{(2l-1)z}{r^2} S_{l-1,l-1}. \quad (\text{B11})$$

- (4) More generally, for $0 \leq m < l-1$ the terms are calculated using the three-term recurrence

$$C_{l,m} = \frac{(2l-1)z}{r^2} C_{l-1,m} - \frac{(l+m-1)(l-m-1)}{r^2} C_{l-2,m}, \quad (\text{B12})$$

$$S_{l,m} = \frac{(2l-1)z}{r^2} S_{l-1,m} - \frac{(l+m-1)(l-m-1)}{r^2} S_{l-2,m}. \quad (\text{B13})$$

The values of $C_{l,m}, S_{l,m}$ for negative values of m are related to their positive counterparts up to a sign, so do not need to be computed or stored.

The fast inverse square root [38] can be used to compute $1/r = \|\mathbf{r}_i - \mathbf{c}_a\|^{-1}$ with complexity comparable to four multiplications [5]. In turn, that can be used to compute $1/r^2, x/r^2, y/r^2$, and z/r^2 with four multiplications. Given that these quantities have been calculated, and ignoring for the moment the multiplication by classically chosen numbers such as $2l-1$, we have the following accounting of the number of multiplications.

- (i) Step 2 above is performed $\mathcal{P}$ times, and each has two multiplications, for $2\mathcal{P}$ multiplications.
(ii) Step 3 above is also performed $\mathcal{P}$ times and has two multiplications, for a total of $2\mathcal{P}$ multiplications.

- (iii) Step 4 above is performed $\mathcal{P}(\mathcal{P}-1)/2$ times, and each has four multiplications, for a total of $2\mathcal{P}(\mathcal{P}-1)$.

Together there is a total of $2\mathcal{P}(\mathcal{P}+1)$ multiplications.

The multiplications by $2l-1$ and $(l+m-1)(l-m-1)$ correspond to what is known as the multiple constant multiplication problem [65], and cost only a single addition each. For example, for $l=2$, $2l-1=3=2+1$, so multiplication by $2l-1$ can be obtained by one addition between the multiplicand and the bit-shifted multiplicand. Then for $l=4$, $2l-1=7=4+3$, so the result can be obtained from an addition between the previous result and the bit-shifted multiplicand. Similarly, all other results can be obtained either by a single addition directly or a single addition between one of the other results and the bit-shifted multiplicand. For the multiplications by $(l+m-1)(l-m-1)$, most can be obtained in this way, but others can be obtained by bit-shifted copies of other products. Therefore, we can ignore this cost in comparison to the other products.

For the total cost of the MP procedure, we lastly need to multiply by the coefficients $M_{lm}(a)$, which gives another $(\mathcal{P}+1)^2$ multiplications. As a result, the total number of multiplications is $3\mathcal{P}^2 + 4\mathcal{P} + 9$. In this case there is *no* extra factor of 2 for uncomputation, because the contribution to the potential may be used to apply a phase factor for the simulation, and does not need to be added into a total potential. Multiplying by a factor of 448 for the boxes in the interaction list with sorts gives $1344\mathcal{P}^2 + 1792\mathcal{P} + 4032$. In this approach the MM translation is still needed, but not the LL translation, giving the number of multiplications per particle

$$\begin{aligned} & \left(1344\mathcal{P}^2 + 1792\mathcal{P} + 4032 + \frac{10}{3}\mathcal{P}^3 + \frac{25}{2}\mathcal{P}^2 + \frac{73}{6}\mathcal{P} + 3 \right) \\ & \times (L-3) \approx \left(\frac{10}{3}\mathcal{P}^3 + 1356\mathcal{P}^2 + 1804\mathcal{P} + 4035 \right) (L-3). \end{aligned} \quad (\text{B14})$$

For $\mathcal{P} = 18$, this is less than $1/20$ the multiplication count of the quantum FMM in Eq. (B4) and even less than $1/4$ the multiplication count of the straightforward translation of the Greengard and Rokhlin algorithm given in Eq. (B3).

For the example of $L = 5$, this would give about 660,000 multiplications per particle for $\mathcal{P} = 9$, or about 2.5 million for $\mathcal{P} = 18$. In comparison, the explicit summation needs approximately 4 multiplications for each of $\eta(\eta-1)/2$ pairwise potentials, for a total of about 2η multiplications per particle. This would give a threshold for speedup over direct summation of about 330,000 particles for $\mathcal{P} = 9$, or 1.2 million for $\mathcal{P} = 18$. These thresholds are far greater than estimated for the classical

FMM, indicating that further optimization of the quantum implementation would be needed for it to be practical.

In practice, in FMM there is a considerable reduction in the complexity due to using far fewer boxes than particles, which results in it being more efficient than performing calculations using the MP. It may therefore be expected that to achieve performance comparable to the classical FMM, the implementation of the quantum FMM should be modified to only represent information for the maximum number of boxes needed. It may also be advantageous to develop a hybrid scheme with direct summation or the quantum FMM for evenly distributed particles.

-
- [1] E. C. Eklund, A. Tikku, P. Sinnott, W. J. Huggins, G. H. Low, D. W. Berry, and I. Kassal, *End-to-end simulation of chemical dynamics on a quantum computer*, [arXiv:2603.19007](#).
 - [2] 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).
 - [3] 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 Inf* **10**, 130 (2024).
 - [4] 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*, *Nat. Commun.* **14**, 4058 (2023).
 - [5] 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*, *Proc. Natl. Acad. Sci. U.S.A.* **121**, e2317772121 (2024).
 - [6] S. Lloyd, *Universal quantum simulators*, *Science* **273**, 1073 (1996).
 - [7] N. Wiebe and A. M. Childs, *Hamiltonian simulation using linear combinations of unitary operations*, *Quantum Inf. Comput.* **12**, 901 (2012).
 - [8] 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, 10.1145/2591796.2591854.
 - [9] D. W. Berry and A. M. Childs, *Black-box Hamiltonian simulation and unitary implementation*, *Quantum Inf. Comput.* **12**, 0029 (2012).
 - [10] 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).
 - [11] G. H. Low and I. L. Chuang, *Optimal hamiltonian simulation by quantum signal processing*, *Phys. Rev. Lett.* **118**, 010501 (2017).
 - [12] G. H. Low and I. L. Chuang, *Hamiltonian simulation by qubitization*, *Quantum* **3**, 163 (2019).
 - [13] 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).
 - [14] 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 Inf. Comput.* **25**, 1 (2025).
 - [15] G. H. Low and N. Wiebe, *Hamiltonian simulation in the interaction picture*, [arXiv:1805.00675](#).
 - [16] T. Stetina and N. Wiebe, *First-quantized quantum simulation of non-relativistic QED with emergent topologically protected Coulomb interactions*, [arXiv: 2508.19343](#).
 - [17] I. Kassal, S. P. Jordan, P. J. Love, M. Mohseni, and A. Aspuru-Guzik, *Polynomial-time quantum algorithm for the simulation of chemical dynamics*, *Proc. Natl. Acad. Sci. U.S.A.* **105**, 18681 (2008).
 - [18] G. H. Low, Y. Su, Y. Tong, and M. C. Tran, *Complexity of implementing Trotter steps*, *PRX Quantum* **4**, 020323 (2023).
 - [19] J. Barnes and P. Hut, *A hierarchical $O(N \log N)$ force-calculation algorithm*, *Nature (London)* **324**, 446 (1986).
 - [20] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, *J. Comput. Phys.* **60**, 187 (1985).
 - [21] J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, *SIAM J. Sci. Stat. Comput.* **9**, 669 (1988).
 - [22] A. M. Childs, J. Leng, T. Li, J.-P. Liu, and C. Zhang, *Quantum simulation of real-space dynamics*, *Quantum* **6**, 860 (2022).
 - [23] S.-T. Cheng and C.-Y. Wang, *Quantum switching and quantum merge sorting*, *IEEE Trans. Circ. Syst. I: Regular Papers* **53**, 316 (2006).
 - [24] R. Beals, S. Brierley, O. Gray, A. W. Harrow, S. Kutin, N. Linden, D. Shepherd, and M. Stather, *Efficient distributed quantum computing*, *Proc. R. Soc. A* **469**, 20120686 (2013).
 - [25] A. Yesypenko, C. Chen, and P.-G. Martinsson, *A simplified fast multipole method based on strong recursive skeletonization*, *J. Comput. Phys.* **524**, 113707 (2025).
 - [26] R. E. Angulo and O. Hahn, *Large-scale dark matter simulations*, *Liv. Rev. Comput. Astrophys.* **8**, 1 (2022).
 - [27] I. Fukuda and H. Nakamura, *Non-Ewald methods for evaluating the electrostatic interactions of charge systems: Similarity and difference*, *Biophys. Rev.* **14**, 1315 (2022).
 - [28] Q. Wang, *A hybrid fast multipole method for cosmological N -body simulations*, *Res. Astron. Astrophys.* **21**, 003 (2021).
 - [29] N. Y. Gnedin, *Hierarchical particle mesh: An FFT-accelerated fast multipole method*, *Astrophys. J. Suppl. Ser.* **243**, 19 (2019).
 - [30] B. Stenqvist, V. Aspelin, and M. Lund, *Generalized moment correction for long-ranged electrostatics*, *J. Chem. Theory Comput.* **16**, 3737 (2020).
 - [31] J. Liang, L. Lu, A. Barnett, L. Greengard, and S. Jiang, *Accelerating fast Ewald summation with prolates for molecular dynamics simulations*, *Nat. Commun.* **17**, 6681 (2026).
 - [32] L. Greengard and V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, *Acta Numer.* **6**, 229 (1997).

- [33] Y. Su, H.-Y. Huang, and E. T. Campbell, *Nearly tight Trotterization of interacting electrons*, *Quantum* **5**, 495 (2021).
- [34] R. Babbush, D. W. Berry, J. R. McClean, and H. Neven, *Quantum simulation of chemistry with sublinear scaling in basis size*, *npj Quantum Inf.* **5**, 92 (2019).
- [35] R. Babbush, N. Wiebe, J. McClean, J. McClain, H. Neven, and G. K.-L. Chan, *Low-depth quantum simulation of materials*, *Phys. Rev. X* **8**, 011044 (2018).
- [36] 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*, *Phys. Rev. X* **8**, 041015 (2018).
- [37] 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).
- [38] T. Häner, M. Roetteler, and K. M. Svore, *Optimizing quantum circuits for arithmetic*, [arXiv:1805.12445](https://arxiv.org/abs/1805.12445).
- [39] 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, [10.1109/ICDE.2001.914876](https://doi.org/10.1109/ICDE.2001.914876).
- [40] K. Barkalov, A. Shtanyuk, and A. Sysoyev, *A fast k NN algorithm using multiple space-filling curves*, *Entropy* **24**, 767 (2022).
- [41] S. Li, L. Simons, J. B. Pakaravoor, F. Abbasinejad, J. D. Owens, and N. Amenta, *k ANN 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), pp. 39–47.
- [42] A. Grüneis, J. J. Shepherd, A. Alavi, D. P. Tew, and G. H. Booth, *Explicitly correlated plane waves: Accelerating convergence in periodic wavefunction expansions*, *J. Chem. Phys.* **139**, 84112 (2013).
- [43] C. Hättig, W. Klopper, A. Köhn, and D. P. Tew, *Explicitly correlated electrons in molecules*, *Chem. Rev.* **112**, 4 (2011).
- [44] J. Harl and G. Kresse, *Cohesive energy curves for noble gas solids calculated by adiabatic connection fluctuation-dissipation theory*, *Phys. Rev. B* **77**, 45136 (2008).
- [45] 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*, *Phys Rev B* **86**, 035111 (2012).
- [46] T. Helgaker, W. Klopper, H. Koch, and J. Noga, *Basis-set convergence of correlated calculations on water*, *J. Chem. Phys.* **106**, 9639 (1998).
- [47] W. Klopper, *Limiting values for Møller-Plesset second-order correlation energies of polyatomic systems: A benchmark study on Ne, HF, H₂O, N₂, and He...He*, *J. Chem. Phys.* **102**, 6168 (1995).
- [48] A. Halkier, T. Helgaker, P. Jørgensen, W. Klopper, H. Koch, J. Olsen, and A. K. Wilson, *Basis-set convergence in correlated calculations on Ne, N₂, and H₂*, *Chem. Phys. Lett.* **286**, 243 (1998).
- [49] P. Panteleev and G. Kalachev, *Degenerate quantum LDPC codes with good finite length performance*, *Quantum* **5**, 585 (2021).
- [50] M. Cain, Q. Xu, R. King, L. R. B. Picard, H. Levine, M. Endres, J. Preskill, H.-Y. Huang, and D. Bluvstein, *Shor's algorithm is possible with as few as 10,000 reconfigurable atomic qubits*, [arXiv:2603.28627](https://arxiv.org/abs/2603.28627).
- [51] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, *Elucidating reaction mechanisms on quantum computers*, *Proc. Natl. Acad. Sci. U.S.A.* **114**, 7555 (2017).
- [52] L. Greengard and V. Rokhlin, *On the efficient implementation of the fast multipole algorithm*, Yale University, Department of Computer Science, Technical Report, 1988.
- [53] C. Qian, M. Wang, and A. C. Yucel, *Compression of far-fields in the fast multipole method via Tucker decomposition*, *IEEE Trans. Antennas Propag.* **69**, 6660 (2021).
- [54] A. J. Hesford and R. C. Waag, *Reduced-rank approximations to the far-field transform in the gridded fast multipole method*, *J. Comput. Phys.* **230**, 3656 (2011).
- [55] D. Potter, J. Stadel, and R. Teyssier, *PKDGRAV3: Beyond trillion particle cosmological simulations for the next era of galaxy surveys*, *Comput. Astrophys. Cosmol.* **4**, 2 (2017).
- [56] K. E. Schmidt and M. A. Lee, *Implementing the fast multipole method in three dimensions*, *J. Stat. Phys.* **63**, 1223 (1991).
- [57] K. Schmidt and M. A. Lee, *Multipole Ewald sums for the fast multipole method*, *J. Stat. Phys.* **89**, 411 (1997).
- [58] A. D. Baczewski and B. Shanker, *An $\mathcal{O}(N)$ method for rapidly computing periodic potentials using accelerated Cartesian expansions*, [arXiv:1107.3069](https://arxiv.org/abs/1107.3069).
- [59] M. Fenn and G. Steidl, *FMM and H-matrices: A short introduction to the basic idea* (2002), <https://page.math.tu-berlin.de/~steidl/PDFs/FenSte02.pdf>.
- [60] B. Shanker and H. Huang, *Accelerated Cartesian expansions—A fast method for computing of potentials of the form $R^{-\nu}$ for all real ν* , *J. Comput. Phys.* **226**, 732 (2007).
- [61] M. T. Goodrich, *Zig-zag sort: a simple deterministic data-oblivious sorting algorithm running in $O(n \log n)$ time*, in *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, STOC'14* (Association for Computing Machinery, New York, NY, USA, 2014), pp. 684–693, [10.1145/2591796.2591830](https://doi.org/10.1145/2591796.2591830).
- [62] K. E. Batcher, *Sorting networks and their applications*, in *Proceedings of the Spring Joint Computer Conference, AFIPS '68 (Spring)* (Association for Computing Machinery, New York, NY, USA, 1968), pp. 307–314, [10.1145/1468075.1468121](https://doi.org/10.1145/1468075.1468121).
- [63] P. Niroula and Y. Nam, *A quantum algorithm for string matching*, *npj Quantum Inf.* **7**, 37 (2021).
- [64] M. Perez-Saborid, *The coordinate-free approach to spherical harmonics*, [arXiv:0806.3367](https://arxiv.org/abs/0806.3367).
- [65] Y. Voronenko and M. Püschel, *Multiplierless multiple constant multiplication*, *ACM Trans. Algorithms* **3**, 11 (2007).