Exponentially more precise quantum simulation of fermions in second quantization

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
29 October 2015

REVISED
28 January 2016

ACCEPTED FOR PUBLICATION
1 March 2016

PUBLISHED
24 March 2016

PAPER

Exponentially more precise quantum simulation of fermions in second quantization

Ryan Babbush[1,2], Dominic W Berry[3], Ian D Kivlichan[1,4], Annie Y Wei[1], Peter J Love[5] and Alán Aspuru-Guzik[1]

[1] Department of Chemistry and Chemical Biology, Harvard University, Cambridge, MA 02138, USA
[2] Google, Venice, CA 90291, USA
[3] Department of Physics and Astronomy, Macquarie University, Sydney, NSW 2109, Australia
[4] Department of Physics, Harvard University, Cambridge, MA 02138, USA
[5] Department of Physics and Astronomy, Tufts University, Medford, MA 02155, USA

E-mail: ryanbabbush@gmail.com, dominic.berry@mq.edu.au **and** aspuru@chemistry.harvard.edu

Keywords: quantum algorithms, quantum simulation, electronic structure theory

Abstract

We introduce novel algorithms for the quantum simulation of fermionic systems which are dramatically more efficient than those based on the Lie–Trotter–Suzuki decomposition. We present the first application of a general technique for simulating Hamiltonian evolution using a truncated Taylor series to obtain logarithmic scaling with the inverse of the desired precision. The key difficulty in applying algorithms for general sparse Hamiltonian simulation to fermionic simulation is that a query, corresponding to computation of an entry of the Hamiltonian, is costly to compute. This means that the gate complexity would be much higher than quantified by the query complexity. We solve this problem with a novel quantum algorithm for on-the-fly computation of integrals that is exponentially faster than classical sampling. While the approaches presented here are readily applicable to a wide class of fermionic models, we focus on quantum chemistry simulation in second quantization, perhaps the most studied application of Hamiltonian simulation. Our central result is an algorithm for simulating an N spin–orbital system that requires $\widetilde{\mathcal{O}}(N^5 t)$ gates. This approach is exponentially faster in the inverse precision and at least cubically faster in N than all previous approaches to chemistry simulation in the literature.

1. Introduction

As small, fault-tolerant quantum computers come increasingly close to viability [1–4] there has been substantial renewed interest in quantum simulating chemistry due to the low qubit requirements and industrial importance of the electronic structure problem. A recent series of papers tried to estimate the resources required to quantum simulate a small but classically intractable molecule [5–9]. Although qubit requirements seem modest, initial predictions of the time required were daunting. Using arbitrarily high-order Trotter formulas, the tightest known bound[6] on the gate count of the second quantized, Trotter-based quantum simulation of chemistry is $\widetilde{\mathcal{O}}(N^8 t/\epsilon^{o(1)})$ [10, 11], where ϵ is the precision required and N is the number of spin–orbitals. However, using significantly more practical Trotter decompositions, the best known gate complexity for this quantum algorithm is $\widetilde{\mathcal{O}}(N^9\sqrt{t^3/\epsilon})$ [6].

Fortunately, numerics indicated that the average circuit depth for real molecules may be closer to $\widetilde{\mathcal{O}}(N^6\sqrt{t^3/\epsilon})$ [7], or $\widetilde{\mathcal{O}}(Z^3N^4\sqrt{t^3/\epsilon})$ [9] when only trying to simulate ground states, where Z is the largest nuclear charge for the molecule. While this improved scaling restores hope that fault-tolerant devices will have an impact on some classically intractable chemistry problems, the Trotter-based quantum simulation of large

[6] We use the typical computer science convention that $f \in \Theta(g)$, for any functions f and g, if f is asymptotically upper and lower bounded by multiples of g, $\mathcal{O}$ indicates an asymptotic upper bound, $\widetilde{\mathcal{O}}$ indicates an asymptotic upper bound up to polylogarithmic factors, Ω indicates the asymptotic lower bound and $f \in o(g)$ implies $f/g \to 0$ in the asymptotic limit.

(e.g. $N > 500$) molecules still seems prohibitively costly [9, 12, 13]. This limitation would preclude simulations of many important molecular systems, such as those involved in biological nitrogen fixation and high-T_c superconductivity [12, 13].

The canonical quantum algorithm for quantum chemistry, based on the Trotter–Suzuki decomposition which was first applied for universal quantum simulation in [14, 15], was introduced nearly one decade ago [16]. This approach was later refined for implementation with a set of universal quantum gates in [17]. With the exception of the adiabatic algorithm described in [18] and a classical variational optimization strategy making use of a quantum wavefunction ansatz described in [19–21], all prior quantum algorithms for chemistry have been based on Trotterization [20, 22–28].

Trotter–Suzuki approaches were also applied to simulation of evolution under sparse Hamiltonians with the entries given by an oracle [29, 30]. A related problem is the simulation of continuous query algorithms; in 2009, Cleve *et al* showed how to achieve such simulation with exponentially fewer discrete queries than Trotterization in terms of $1/\epsilon$ [31]. The algorithm of [31] still required a number of ancilla qubits that scaled polynomially in $1/\epsilon$, but this limitation was overcome in [32] which demonstrated that the ancilla register in [31] could be compressed into exponentially fewer qubits. In [33, 34], Berry *et al* combined the results of [29–32] to show exponentially more precise sparse Hamiltonian simulation techniques. A major contribution of [33] was to use oblivious amplitude amplification to make the algorithm from [31, 32] deterministic, whereas prior versions had relied on probabilistic measurement of ancilla qubits. An improvement introduced in [34] was to show how to simulate arbitrary Hamiltonians using queries that are not self-inverse (a requirement of the procedure in [33]). We focus on the methodology of [34] which is relatively self-contained.

The algorithm of [34] approximates the propagator using a Taylor series expansion rather than the Trotter–Suzuki decomposition. By dividing the desired evolution into a number of simulation segments proportional to the Hamiltonian norm, one can truncate the Taylor series at an order which scales logarithmically in the inverse of the desired precision [34]. The truncated Taylor series must be expressed as a weighted sum of unitary operators. To simulate the action of this operator, one first initializes the system along with an ancilla register that indexes all terms in the Taylor series sum. The ancilla register is then put in a superposition state with amplitudes proportional to the coefficients of terms in the Taylor series sum. Next, an operator is applied to the system which coherently executes a single term in the Taylor series sum that is selected according to the ancilla register in superposition. Finally, by applying the inverse of the procedure which prepares the ancilla register, one probabilistically simulates evolution under the propagator. The algorithm is made deterministic using an oblivious amplitude amplification procedure from [33].

In the present paper we develop two new algorithms for the application of the Hamiltonians terms, which we refer to as the 'database' algorithm and the 'on-the-fly' algorithm. In the database algorithm, the ancilla register's superposition state is prepared with amplitudes from a precomputed classical database. In the on-the-fly algorithm, those amplitudes are computed and prepared on-the-fly, in a way that is exponentially more precise than classically possible.

2. Overview of results

The simulation procedure described in [34] assumes the ability to represent the Hamiltonian as a weighted sum of unitaries which can be individually applied to a quantum state. Specifically, we must be able to express the simulation Hamiltonian as

$$H = \sum_{\gamma=1}^{\Gamma} W_\gamma H_\gamma, \tag{1}$$

where the W_γ are complex-valued scalars[7], the H_γ are unitary operators and a mechanism is available for selectively applying the H_γ. Using the Jordan–Wigner transformation [35, 36] or the Bravyi–Kitaev transformation [37–39], the second quantized molecular Hamiltonian can be mapped to a sum of $\Gamma \in \mathcal{O}(N^4)$ local Hamiltonians. Since these local Hamiltonians are each a tensor product of Pauli operators multiplied by some coefficient, they automatically satisfy the form of equation (1).

We will need a circuit referred to in [34] as $\textsc{select}(H)$ which is queried within the algorithm such that

$$\textsc{select}(H)\left|\gamma\right\rangle\left|\psi\right\rangle = \left|\gamma\right\rangle H_\gamma\left|\psi\right\rangle. \tag{2}$$

[7] The convention of [34] requires that the W_γ are real, non-negative scalars. This treatment remains general as arbitrary phases can be factored into the H_γ. However, we break with that convention and allow the W_γ to take arbitrary complex values. This is done for pedagogical purposes: so that we may separately describe computation of the H_γ and the W_γ for the chemistry Hamiltonian. Consequentially, our equation (39) differs from the analogous equation in [34] by a complex conjugate operator.

One could construct $\textsc{select}(H)$ by storing all the Pauli strings in a database. However, accessing this data would have time complexity of at least $\Omega(\Gamma)$. Instead, we compute and apply the Pauli strings using $\mathcal{O}(N)$ gates (which can be parallelized to $\mathcal{O}(1)$ circuit depth) by dynamically performing the Jordan–Wigner transformation on the quantum hardware.

The algorithm of [34] also requires an operator that we refer to as $\textsc{prepare}(W)$ which applies the mapping

$$\textsc{prepare}(W)|0\rangle^{\otimes \log \Gamma} = \sqrt{\frac{1}{\Lambda}} \sum_{\gamma=1}^{\Gamma} \sqrt{W_\gamma}|\gamma\rangle, \tag{3}$$

where $\Lambda \equiv \sum_{\gamma=1}^{\Gamma} |W_\gamma| \in \mathcal{O}(N^4)$, is a normalization factor that will turn out to have significant ramifications for the algorithm complexity. In the first of two algorithms discussed in this paper, we implement $\textsc{prepare}(W)$ using a database via a sequence of totally controlled rotations at cost $\mathcal{O}(\Gamma)$. Because our first approach uses a database to store classically precomputed values of W_γ in order to implement $\textsc{prepare}(W)$, we refer to the first algorithm as the 'database' algorithm.

While we suggest a different strategy in section 3, a database could also be used to construct $\textsc{select}(H)$. That is, a controlled operation is performed which applies H_1 if $\gamma = 1$, followed by a controlled operation which performs H_2 if $\gamma = 2$, and so forth. This would result in a slightly higher gate count than $\textsc{prepare}(W)$, because each of the Γ controlled operations must act on $\mathcal{O}(\log N)$ qubits even if the Bravyi–Kitaev transformation is used. Nevertheless, this might represent a simpler solution than our construction of $\textsc{select}(H)$ for early experimental implementations.

Our second algorithm involves modifications to the algorithm of [34] which allows us to avoid some of this overhead. We exploit the fact that the chemistry Hamiltonian is easy to express as a special case of equation (1) in which the coefficients are defined by integrals such as

$$W_\gamma = \int_{\mathcal{Z}} w_\gamma(\vec{z})\, \mathrm{d}\vec{z}, \tag{4}$$

where the integrand $w_\gamma(\vec{z})$ represents a scalar-valued function of the vector $\vec{z}$, which is an element of the integration domain $\mathcal{Z}$. Because our approach involves computing integrals on-the-fly, we refer to the second algorithm as the 'on-the-fly' algorithm. We begin by numerically approximating the integrals as finite Riemann sums such as

$$W_\gamma \approx \frac{\mathcal{V}}{\mu} \sum_{\rho=1}^{\mu} w_\gamma(\vec{z}_\rho), \tag{5}$$

where $\vec{z}_\rho$ is a point in the integration domain at grid point ρ. Equation (5) represents a discretization of the integral in equation (4) using μ grid points where the domain of the integral, denoted as $\mathcal{Z}$, has been truncated to have total volume $\mathcal{V}$. This truncation is possible because the functions $w_\gamma(\vec{z})$ can be chosen to decay exponentially over the integration domain for the molecular systems usually studied in chemistry. Note that this might not be true for other systems, such as conducting metals.

Our algorithm is effectively able to numerically compute this integral with complexity logarithmic in the number of grid points. It might be thought that this is impossible, because methods of evaluating numerical integrals on quantum computers normally only give a square-root speedup over classical Monte-Carlo algorithms [40]. The difference here is that we do not output the value of the integral. The value of the integral is only used to control the weight of a term in the Hamiltonian under which the state evolves.

We construct a circuit which computes the values of $w_\gamma(\vec{z}_\rho)$ for the quantum chemistry Hamiltonian with $\widetilde{\mathcal{O}}(N)$ gates. We call this circuit $\textsc{sample}(w)$ and define it by its action

$$\textsc{sample}(w)|\gamma\rangle|\rho\rangle|0\rangle^{\otimes \log M} = |\gamma\rangle|\rho\rangle|\widetilde{w}_\gamma(\vec{z}_\rho)\rangle, \tag{6}$$

where $\widetilde{w}_\gamma(\vec{z}_\rho)$ is the binary representation of $w_\gamma(\vec{z}_\rho)$ using $\log M$ qubits.

By expanding the W_γ in equation (1) in terms of the easily computed $w_\gamma(\vec{z})$ as in equation (5), we are able to compute analogous amplitudes to those in equation (3) in an efficient fashion. Thus, we no longer need the database that characterizes that algorithm. State preparation where the state coefficients can be computed on the quantum computer is more efficient than when they are stored on, and accessed from, a database [41]. The worst-case complexity is the square root of the dimension (here it would be $\mathcal{O}(\sqrt{\Gamma\mu})$), whereas the database state preparation has complexity linear in the dimension (which is $\mathcal{O}(\Gamma)$ for W_γ). Here this would not be an improvement, as we have increased the dimension in the discretization of the integral.

However, the worst-case complexity is only if the amplitudes can take arbitrary values (as this would enable a search algorithm, where the square root of the dimension is optimal [42]). If the amplitudes differ only by phases, the complexity of the state preparation is logarithmic in the dimension. We therefore decompose each $w_\gamma(\vec{z})$ into a sum of terms which differ only by a sign, $w_{\gamma,m}(\vec{z})$. Then, although the dimension is increased, the complexity of the state preparation is reduced. In turn, we can express the Hamiltonian as a sum of unitaries weighted by identical amplitudes which differ only by an easily computed sign

$$H = \frac{\zeta\mathcal{V}}{\mu}\sum_{\gamma=1}^{\Gamma}\sum_{m=1}^{M}\sum_{\rho=1}^{\mu} w_{\gamma,m}(\vec{z}_\rho)H_\gamma. \tag{7}$$

As discussed above, the state preparation needed can be performed much more efficiently because the amplitudes are now identical up to a phase. By making a single query to $\textsc{sample}(w)$ and then performing phase-kickback we can implement the operator $\textsc{prepare}(w)$ whose action is

$$\textsc{prepare}(w)|0\rangle^{\otimes\log(L)} = \sqrt{\frac{1}{\lambda}}\sum_{\ell=1}^{L}\sqrt{\frac{\zeta\mathcal{V}}{\mu}w_{\gamma,m}(\vec{z}_\rho)}\,|\ell\rangle, \tag{8}$$

where $|\ell\rangle = |\gamma\rangle|m\rangle|\rho\rangle$, $L \in \Theta(\Gamma M\mu)$ and $\lambda \equiv L\zeta\mathcal{V}/\mu$, is a normalization factor that will turn out to have significant ramifications for the algorithm complexity. Later, we will show that $\lambda \in \widetilde{\mathcal{O}}(N^4)$ and that $\textsc{prepare}(w)$ can be implemented with $\widetilde{\mathcal{O}}(N)$ gate count, the cost of a single query to $\textsc{sample}(w)$.

The database algorithm performs evolution under H for time t by making $\widetilde{\mathcal{O}}(\Lambda t)$ queries to both $\textsc{select}(H)$ and $\textsc{prepare}(W)$. Because $\textsc{prepare}(W)$ requires $\mathcal{O}(\Gamma)=\mathcal{O}(N^4)$ gates, the overall gate count of this approach scales as $\widetilde{\mathcal{O}}(N^4\Lambda t)$. To avoid the overhead from $\textsc{prepare}(W)$, our on-the-fly algorithm exploits a modified version of the truncated Taylor series algorithm which allows for the same evolution by making $\widetilde{\mathcal{O}}(\lambda t)$ queries to $\textsc{select}(H)$ and $\textsc{prepare}(w)$. As $\textsc{prepare}(w)$ requires $\widetilde{\mathcal{O}}(N)$ gates, the gate count for our on-the-fly algorithm scales as $\widetilde{\mathcal{O}}(N\lambda t)$.

The paper is outlined as follows. In section 3 we introduce the second quantized encoding of the wavefunction and construct $\textsc{select}(H)$. In section 4 we review the procedure in [34] to demonstrate our database algorithm which uses $\textsc{select}(H)$ and $\textsc{prepare}(W)$ to perform a quantum simulation which is exponentially more precise than Trotterization. In section 5 we show that one can modify the procedure in [34] to allow for essentially the same result while simultaneously computing the integrals on-the-fly, and show how to implement $\textsc{prepare}(w)$ so as to compute the integrals on-the-fly. In section 6 we bound the errors on the integrals by analyzing the integrands. In section 7 we discuss applications of these results and future research directions.

3. The Hamiltonian oracle

The molecular electronic structure Hamiltonian describes electrons interacting in a fixed nuclear potential. Using atomic units in which the electron mass, electron charge, Coulomb's constant and $\hbar$ are unity we may write the electronic Hamiltonian as

$$H = -\sum_i \frac{\nabla_{\vec{r}_i}^2}{2} - \sum_{i,j}\frac{Z_i}{|\vec{R}_i - \vec{r}_j|} + \sum_{i,j>i}\frac{1}{|\vec{r}_i - \vec{r}_j|}, \tag{9}$$

where $\vec{R}_i$ are the nuclei coordinates, Z_i are the nuclear charges, and $\vec{r}_i$ are the electron coordinates [43]. We represent the system in a basis of N single-particle spin–orbital functions usually obtained as the solution to a classical mean-field treatment such as Hartree–Fock [43]. Throughout this paper, $\varphi_i(\vec{r}_j)$ denotes the ith spin–orbital occupied by the jth electron which is parameterized in terms of spatial degrees of freedom $\vec{r}_j$.

In second quantization, antisymmetry is enforced by the operators whereas in first quantization antisymmetry is explicitly in the wavefunction. The second quantized representation of equation (9) is

$$H = \sum_{ij} h_{ij}a_i^\dagger a_j + \frac{1}{2}\sum_{ijk\ell} h_{ijk\ell}a_i^\dagger a_j^\dagger a_k a_\ell, \tag{10}$$

where the one-electron and two-electron integrals are

$$h_{ij} = \int \varphi_i^*(\vec{r})\left(-\frac{\nabla^2}{2} - \sum_q \frac{Z_q}{|\vec{R}_q - \vec{r}|}\right)\varphi_j(\vec{r})\mathrm{d}\vec{r}, \tag{11}$$

$$h_{ijk\ell} = \int \frac{\varphi_i^*(\vec{r}_1)\varphi_j^*(\vec{r}_2)\varphi_\ell(\vec{r}_1)\varphi_k(\vec{r}_2)}{|\vec{r}_1 - \vec{r}_2|}\,\mathrm{d}\vec{r}_1\,\mathrm{d}\vec{r}_2. \tag{12}$$

The operators $a_i^\dagger$ and a_j in equation (10) obey the fermionic anti-commutation relations

$$\{a_i^\dagger, a_j\} = \delta_{ij}, \qquad \{a_i^\dagger, a_j^\dagger\} = \{a_i, a_j\} = 0. \tag{13}$$

In general, the Hamiltonian in equation (10) contains $\mathcal{O}(N^4)$ terms, except in certain limits of very large molecules where use of a local basis and truncation of terms lead to scaling on the order of $\widetilde{\mathcal{O}}(N^2)$ [8]. The spatial encoding of equation (10) requires $\Theta(N)$ qubits, one for each spin–orbital.

While fermions are antisymmetric, indistinguishable particles, qubits are distinguishable and have no special symmetries. Accordingly, in order to construct the operator SELECT(H), which applies terms in the second quantized Hamiltonian to qubits as in equation (2), we will need a mechanism for mapping the fermionic raising and lowering operators in equation (10) to operators which act on qubits. Operators which raise or lower the state of a qubit are trivial to represent using Pauli matrices

$$\sigma_j^+ = |1\rangle\langle 0| = \frac{1}{2}(\sigma_j^x - \mathrm{i}\,\sigma_j^y), \tag{14}$$

$$\sigma_j^- = |0\rangle\langle 1| = \frac{1}{2}(\sigma_j^x + \mathrm{i}\,\sigma_j^y). \tag{15}$$

Throughout this paper, σ_j^x, σ_j^y and σ_j^z denote Pauli matrices acting on the jth tensor factor. However, these qubit raising and lowering operators do not satisfy the fermionic anti-commutation relations in equation (13). To enforce this requirement we can apply either the Jordan–Wigner transformation [35, 36] or the Bravyi–Kitaev transformation [37–39].

The action of $a_j^\dagger$ or a_j must also introduce a phase to the wavefunction which depends on the parity (i.e. sum modulo 2) of the occupancies of all orbitals with index less than j [38]. If $f_j \in \{0, 1\}$ denotes the occupancy of orbital j then

$$a_j^\dagger|f_N\cdots f_{j+1}\,0\,f_{j-1}\cdots f_1\rangle = (-1)^{\sum_{s=1}^{j-1} f_s}|f_N\cdots f_{j+1}\,1\,f_{j-1}\cdots f_1\rangle, \tag{16}$$

$$a_j|f_N\cdots f_{j+1}\,1\,f_{j-1}\cdots f_1\rangle = (-1)^{\sum_{s=1}^{j-1} f_s}|f_N\cdots f_{j+1}\,0\,f_{j-1}\cdots f_1\rangle, \tag{17}$$

$$a_j^\dagger|f_N\cdots f_{j+1}\,1\,f_{j-1}\cdots f_1\rangle = 0, \tag{18}$$

$$a_j|f_N\cdots f_{j+1}\,0\,f_{j-1}\cdots f_1\rangle = 0. \tag{19}$$

In general, two pieces of information are needed in order to make sure the qubit encoding of the fermionic state picks up the correct phase: the occupancy of the state and the parity of the occupancy numbers up to j. The Jordan–Wigner transformation maps the occupancy of spin–orbital j directly into the state of qubit j. Thus, in the Jordan–Wigner transformation, occupancy information is stored locally. However, in order to measure the parity of the state in this representation, one needs to measure the occupancies of all orbitals less than j. Because of this, the Jordan–Wigner transformed operators are N-local, which means that some of the Jordan–Wigner transformed operators are tensor products of up to N Pauli operators. The Jordan–Wigner transformed operators are

$$a_j^\dagger \equiv \sigma_j^+ \bigotimes_{s=1}^{j-1} \sigma_s^z = \frac{1}{2}(\sigma_j^x - \mathrm{i}\,\sigma_j^y) \otimes \sigma_{j-1}^z \cdots \otimes \sigma_1^z, \tag{20}$$

$$a_j \equiv \sigma_j^- \bigotimes_{s=1}^{j-1} \sigma_s^z = \frac{1}{2}(\sigma_j^x + \mathrm{i}\,\sigma_j^y) \otimes \sigma_{j-1}^z \cdots \otimes \sigma_1^z. \tag{21}$$

It would be convenient if we could construct SELECT(H) by applying the Jordan–Wigner transform and acting on the quantum state, one spin–orbital index at a time. For instance, SELECT(H) might control the application of a fermionic operator as follows

$$\begin{aligned} |ijk\ell\rangle|\psi\rangle &\mapsto |ijk\ell\rangle a_\ell|\psi\rangle \\ &\mapsto |ijk\ell\rangle a_k a_\ell|\psi\rangle \\ &\mapsto |ijk\ell\rangle a_j^\dagger a_k a_\ell|\psi\rangle \\ &\mapsto |ijk\ell\rangle a_i^\dagger a_j^\dagger a_k a_\ell|\psi\rangle. \end{aligned} \tag{22}$$

However, the operators $a_j^\dagger$ and a_j are not unitary because the operators σ^+ and σ^- are not unitary. To correct this problem, we add four qubits to the selection register where each of the four qubits indicates whether the σ^x or the $\pm\mathrm{i}\,\sigma^y$ part of the σ^+ and σ^- operators should be applied for each of the four fermionic operators in a string such as $a_i^\dagger a_j^\dagger a_k a_\ell$. For ease of exposition, we define new fermionic operators which are unitary, $a_{j,q}^\dagger$ and $a_{j,q}$, where $q \in \{0, 1\}$

$$a_{j,0}^\dagger \equiv \sigma_j^x \bigotimes_{s=1}^{j-1} \sigma_s^z, \qquad a_{j,1}^\dagger \equiv -\mathrm{i}\,\sigma_j^y \bigotimes_{s=1}^{j-1} \sigma_s^z, \tag{23}$$

$$a_{j,0} \equiv \sigma_j^x \bigotimes_{s=1}^{j-1} \sigma_s^z, \qquad a_{j,1} \equiv \mathrm{i}\,\sigma_j^y \bigotimes_{s=1}^{j-1} \sigma_s^z. \tag{24}$$

We use these definitions to rewrite the Hamiltonian in equation (10) so that it is explicitly a weighted sum of unitary Pauli products of the form we require in equation (1)

$$H = \sum_{q_1 q_2}\sum_{ij}\frac{h_{ij}}{4}a_{i,q_1}^{\dagger}a_{j,q_2} + \sum_{q_1 q_2 q_3 q_4}\sum_{ijk\ell}\frac{h_{ijk\ell}}{32}a_{i,q_1}^{\dagger}a_{j,q_2}^{\dagger}a_{k,q_3}a_{\ell,q_4}. \tag{25}$$

Inspection reveals that applying the transformations in equations (23) and (24) to equation (25) gives the same expression as applying the transformations in equations (20) and (21) to equation (10). By removing factors of $1/2$ from both transformation operators and instead placing them in equation (25), we obtain transformation operators that are always unitary tensor products of Pauli operators.

Accordingly, we can implement SELECT(H) in the spirit of equation (22) by using four additional qubits and the transformation operators in equations (23) and (24) so that

$$\begin{aligned} |ijk\ell\rangle|q_1q_2q_3q_4\rangle|\psi\rangle &\mapsto |ijk\ell\rangle|q_1q_2q_3q_4\rangle a_{\ell,q_4}|\psi\rangle \\ &\mapsto |ijk\ell\rangle|q_1q_2q_3q_4\rangle a_{k,q_3}a_{\ell,q_4}|\psi\rangle \\ &\mapsto |ijk\ell\rangle|q_1q_2q_3q_4\rangle a_{j,q_2}^{\dagger}a_{k,q_3}a_{\ell,q_4}|\psi\rangle \\ &\mapsto |ijk\ell\rangle|q_1q_2q_3q_4\rangle a_{i,q_1}^{\dagger}a_{j,q_2}^{\dagger}a_{k,q_3}a_{\ell,q_4}|\psi\rangle. \end{aligned} \tag{26}$$

A circuit which implements these operators controlled on the selection register is straightforward to construct. Furthermore, the transformation of the terms can be accomplished in $\mathcal{O}(1)$ time. Because the Jordan–Wigner transformation is N-local, the number of gates required to actually apply the unitaries in SELECT(H) is $\mathcal{O}(N)$. However, the terms in equations (23) and (24) are trivial to apply in parallel so that each query takes $\mathcal{O}(1)$ time.

Whereas the Jordan–Wigner transformation stores occupancy information locally and parity information N-locally, the Bravyi–Kitaev transformation stores both parity and occupancy information in a number of qubits that scales as $\mathcal{O}(\log N)$ [37–39]. For this reason, the operators obtained using the Bravyi–Kitaev basis act on at most $\mathcal{O}(\log N)$ qubits. It might be possible to apply the Bravyi–Kitaev transformation with $\mathcal{O}(\log N)$ gates, which would allow for an implementation of SELECT(H) with $\mathcal{O}(\log N)$ instead of $\mathcal{O}(N)$ gates. However, the Bravyi–Kitaev transformation is much more complicated and this would not change the asymptotic scaling of our complete algorithm. The reason for this is because the total cost will depend on the sum of the gate count of SELECT(H) and the gate count of PREPARE(W) or PREPARE(w), and the latter procedures always require at least $\mathcal{O}(N)$ gates.

4. Simulating Hamiltonian evolution

Using the method of [34], Hamiltonian evolution can be simulated with an exponential improvement in precision over Trotter-based methods by approximating the truncated Taylor series of the time evolution operator $U = \mathrm{e}^{-\mathrm{i}Ht}$. We begin by partitioning the total simulation time t into r segments of time t/r. For each of these r segments we perform a Taylor expansion of the propagator and truncate the series at order K, i.e.

$$\begin{aligned} U_r \equiv \mathrm{e}^{-\mathrm{i}Ht/r} &\approx \sum_{k=0}^{K}\frac{(-\mathrm{i}Ht/r)^k}{k!} \\ &= \sum_{k=0}^{K}\sum_{\gamma_1,\ldots,\gamma_k=1}^{\Gamma}\frac{(-\mathrm{i}t/r)^k}{k!}W_{\gamma_1}\cdots W_{\gamma_k}H_{\gamma_1}\cdots H_{\gamma_k}, \end{aligned} \tag{27}$$

where in the second line we have expanded H as in equation (1). Notice that if we truncate the series at order K, we incur error

$$\mathcal{O}\left(\frac{(\|H\|\ t/r)^{K+1}}{(K+1)!}\right). \tag{28}$$

If we wish for the total simulation to have error less than ϵ, each segment must have error less than ϵ/r. Accordingly, if we set $r \geqslant \|H\|t$ then our total simulation will have error at most ϵ if

$$K \in \mathcal{O}\left(\frac{\log(r/\epsilon)}{\log\log(r/\epsilon)}\right). \tag{29}$$

We now discuss how one can actually implement the truncated evolution operator in equation (27). First note that the sum in equation (27) takes the form

$$\begin{aligned} \widetilde{U} &= \sum_j \beta_j V_j, \qquad j \equiv (k,\ \gamma_1,\ldots,\gamma_k), \\ \beta_j &\equiv \frac{t^k}{r^k k!}W_{\gamma_1}\cdots W_{\gamma_k}, \qquad V_j \equiv (-\mathrm{i})^k H_{\gamma_1}\ldots H_{\gamma_k}, \end{aligned} \tag{30}$$

Table 1. Database algorithm parameters and bounds.

Parameter	Explanation	Bound
Λ	Normalization factor, equation (37)	$\mathcal{O}(N^4)$
r	Number of time segments, equation (40)	$\Lambda t/\ln(2)$
K	Truncation point for Taylor series, equation (29)	$\mathcal{O}\left(\frac{\log(r/\epsilon)}{\log\log(r/\epsilon)}\right)$
Γ	Number of terms in unitary decomposition, equation (1)	$\mathcal{O}(N^4)$
J	Number of ancilla qubits in selection register, equation (31)	$\Theta(K\log\Gamma)$

Table 2. Database algorithm operators and gate counts.

Operator	Purpose	Gate count
$\textsc{select}(H)$	Applies specified terms from decomposition, equation (2)	$\mathcal{O}(N)$
$\textsc{select}(V)$	Applies specified strings of terms, equation (32)	$\mathcal{O}(NK)$
$\textsc{prepare}(W)$	Prepares a superposition of states weighted by coefficients, equation (3)	$\mathcal{O}(\Gamma)$
$\textsc{prepare}(\beta)$	Prepares a superposition of states weighted by coefficients, equation (33)	$\mathcal{O}(K\Gamma)$
$\mathcal{W}$	Probabilistically performs simulation under H for time t/r, equation (39)	$\mathcal{O}(K\Gamma)$
P	Projects system onto $\lvert 0\rangle^{\otimes J}$ state of selection register, equation (42)	$\Theta(K\log\Gamma)$
G	Amplification operator to implement sum of unitaries, equation (43)	$\mathcal{O}(K\Gamma)$
$(PG)^r$	Entire algorithm	$\mathcal{O}(rK\Gamma)$

where the V_j are unitary and $\widetilde{U}$ is close to unitary. Our simulation uses an ancillary 'selection' register $\lvert j\rangle = \lvert k\rangle\lvert\gamma_1\rangle\cdots\lvert\gamma_K\rangle$, where $0 \leqslant k \leqslant K$ and $1 \leqslant \gamma_\upsilon \leqslant \Gamma$ for all υ. We will encode k in unary, which requires $\Theta(K)$ qubits, so that $\lvert k\rangle = \lvert 1^k 0^{K-k}\rangle$. Additionally, we encode each $\lvert\gamma_\upsilon\rangle$ in binary using $\Theta(\log\Gamma)$ qubits. While we need K of the $\lvert\gamma_\upsilon\rangle$ registers, we note that only k will actually be in use for a given value of $\lvert k\rangle$. The total number of ancilla qubits required for the selection register $\lvert j\rangle$, denoted as J, scales as

$$J \in \Theta(K\log\Gamma) = \mathcal{O}\left(\frac{\log(N)\log(r/\epsilon)}{\log\log(r/\epsilon)}\right). \tag{31}$$

By making $\mathcal{O}(K)$ queries to $\textsc{select}(H)$ from section 2, we can implement an operator to apply the V_j which is referred to in [34] as $\textsc{select}(V)$

$$\textsc{select}(V)\lvert j\rangle\lvert\psi\rangle = \lvert j\rangle V_j\lvert\psi\rangle, \tag{32}$$

where the V_j are defined as in equation (30). This is equivalent to k applications of $\textsc{select}(H)$, using each of the $\lvert\gamma_\upsilon\rangle$ registers, together with k multiplications by $-\mathrm{i}$. In order to obtain k applications of $\textsc{select}(H)$, we may perform a controlled form of $\textsc{select}(H)$ K times, with each successive qubit in the unary representation of k as the control. Given that the gate count for $\textsc{select}(H)$ scales as $\mathcal{O}(N)$, we can implement $\textsc{select}(V)$ with $\mathcal{O}(NK)$ gates. Applying the Pauli strings in parallel leads to circuit depths of $\mathcal{O}(1)$ and $\mathcal{O}(K)$, respectively. Table 1 lists relevant parameters along with their bounds in our database algorithm. Table 2 lists relevant operators and their gate counts in our database algorithm.

We will also need an operator that we refer to as $\textsc{prepare}(\beta)$, which initializes a state

$$\textsc{prepare}(\beta)\lvert 0\rangle^{\otimes J} = \sqrt{\frac{1}{s}}\sum_j\sqrt{\beta_j}\lvert j\rangle, \tag{33}$$

where s is a normalization factor. To implement $\textsc{prepare}(\beta)$ we first prepare the state

$$\left(\sum_{k=0}^{K}\frac{(\Lambda t/r)^k}{k!}\right)^{-1/2}\sum_{k=0}^{K}\sqrt{\frac{(\Lambda t/r)^k}{k!}}\lvert k\rangle. \tag{34}$$

Using the convention that $R_y(\theta) \equiv \exp[-\mathrm{i}\,\theta\,\sigma^y/2]$, we apply $R_y(\theta_1)$ to the first qubit of the unary encoding for k followed by $R_y(\theta_k)$ to the kth qubit controlled on qubit $k-1$ for all $k \in [2, K]$ sequentially, where

$$\theta_k \equiv 2\arcsin\left(\sqrt{1 - \frac{(\Lambda t/r)^{k-1}}{(k-1)!}\left(\sum_{q=k}^{K}\frac{(\Lambda t/r)^q}{q!}\right)^{-1}}\right). \tag{35}$$

To each of the K remaining components of the selection register $\lvert\gamma_1\rangle\cdots\lvert\gamma_K\rangle$, we apply $\textsc{prepare}(W)$ once, which acts as

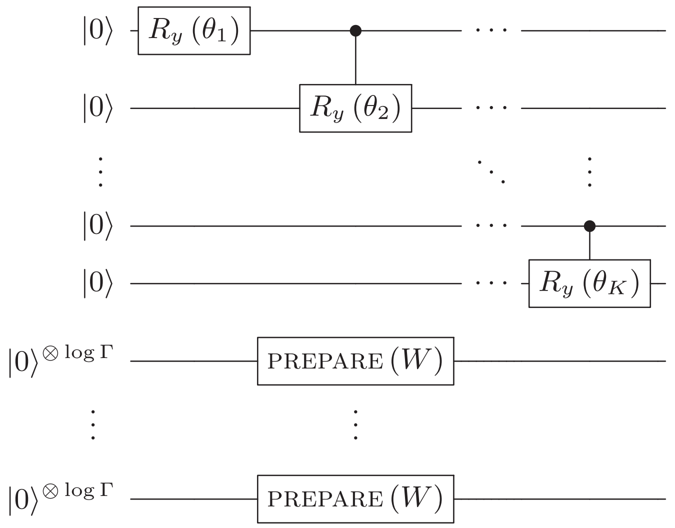

Figure 1. The circuit for PREPARE(β) as described in equation (33). An expression for θ_k is given in equation (35). PREPARE(W) is implemented using a precomputed classical database.

$$\text{PREPARE}(W)|0\rangle^{\otimes \log \Gamma} = \sqrt{\frac{1}{\Lambda}} \sum_{\gamma=1}^{\Gamma} \sqrt{W_\gamma} |\gamma\rangle, \tag{36}$$

where

$$\Lambda \equiv \sum_{\gamma=1}^{\Gamma} |W_\gamma|, \qquad \Lambda \in \mathcal{O}(N^4). \tag{37}$$

In principle, we only need to perform PREPARE(W) k times, because the registers past k are not used. However, it is simpler to perform PREPARE(W) K times, because it does not require control on k.

Using results from [44], PREPARE(W) can be implemented with $\mathcal{O}(\Gamma)$ gates by using a classically precomputed database of the Γ molecular integrals. The gate count for PREPARE(β) thus scales as $\mathcal{O}(K\Gamma)$. However, this construction is naturally parallelized to depth $\mathcal{O}(K + \Gamma)$. A circuit implementing PREPARE(β) is shown in figure 1.

The general strategy for implementing the truncated evolution operator in equation (30) becomes clear if we consider what happens to state $|\psi\rangle$ when we apply PREPARE(β) followed by the operator SELECT(V)

$$\text{SELECT}(V)\text{PREPARE}(\beta)|0\rangle^{\otimes J}|\psi\rangle = \sum_j \sqrt{\frac{\beta_j}{s}} |j\rangle V_j|\psi\rangle. \tag{38}$$

The similarity of this state to the state $\widetilde{U}|\psi\rangle$ motivates the operator

$$\begin{aligned} \mathcal{W} &\equiv (\text{PREPARE}(\beta) \otimes \mathbb{1})^{\top}\text{SELECT}(V)(\text{PREPARE}(\beta) \otimes \mathbb{1}), \\ \mathcal{W}|0\rangle^{\otimes J}|\psi\rangle &= \frac{1}{s}|0\rangle\, \widetilde{U}|\psi\rangle + \sqrt{1 - \frac{1}{s^2}}\,|\Phi\rangle, \end{aligned} \tag{39}$$

where $|\Phi\rangle$ is a state with the ancilla qubits orthogonal to $|0\rangle^{\otimes J}$. Note that in [34], the authors use the convention that all W_γ are positive and phases are incorporated into the operators H_γ. Since we depart from that convention for reasons described in section 2, the second application of PREPARE(β) in equation (39) is the transpose of the first application, in contrast to [34] where the conjugate transpose is used instead. The circuit implementing $\mathcal{W}$ is shown in figure 2.

At this point, we choose the number of segments to be

$$r = \Lambda t/\ln(2). \tag{40}$$

Since $\Lambda \geqslant \|H\|$, our choice of K in equation (29) remains valid. The additional factor of $1/\ln(2)$ is included to satisfy a requirement of oblivious amplitude amplification as described in [33] so that

$$s = \sum_j |\beta_j| = \sum_{k=0}^{K} \frac{1}{k!} \ln(2)^k \approx 2. \tag{41}$$

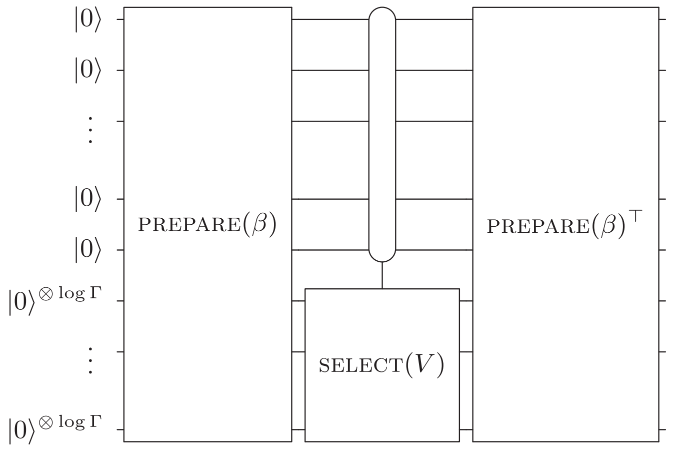

Figure 2. The circuit implementing $\mathcal{W}$. The oval indicates the control register for SELECT(V).

We now define a projection operator P onto the target state, which has support on the empty ancilla register

$$P \equiv (|0\rangle\langle 0|)^{\otimes J} \otimes \mathbb{1},$$

$$P\mathcal{W}|0\rangle^{\otimes J}|\psi\rangle = \frac{1}{s}|0\rangle^{\otimes J}\widetilde{U}|\psi\rangle. \tag{42}$$

We also define the amplification operator

$$G \equiv -\mathcal{W}R\mathcal{W}^{\dagger}R\mathcal{W}, \tag{43}$$

where $R = \mathbb{1} - 2P$ is the reflection operator. With these definitions, we follow the procedure in [34] which uses the oblivious amplitude amplification procedure of [33] to deterministically execute the intended unitary. We use G in conjunction with P to amplify the target state

$$PG|0\rangle\,|\psi\rangle = |0\rangle\left(\frac{3}{s}\widetilde{U} - \frac{4}{s^3}\widetilde{U}\widetilde{U}^{\dagger}\widetilde{U}\right)|\psi\rangle. \tag{44}$$

Recalling the definition of U_r in equation (27), our choices of K in equation (29) and $r = \Lambda t/\ln 2$ imply that

$$\| PG|0\rangle\,|\psi\rangle - |0\rangle\, U_r|\psi\rangle \| \in \mathcal{O}(\epsilon/r), \tag{45}$$

so that the total error from applying oblivious amplitude amplification to all the segments will again be order ϵ. To summarize, the database algorithm is as below.

(1) Express the Hamiltonian as a weighted sum of unitary operators as in equation (1).

(2) Subdivide the simulation time t into $r = \Lambda t/\ln(2)$ segments, where Λ is defined in equation (37).

(3) Expand the evolution for time t/r as a truncated Taylor series U_r, as defined in equation (27). (Steps 1 to 3 are classical pre-processing.)

(4) For each segment perform the following steps.

(a) Prepare a superposition state with amplitudes proportional to $\sqrt{W_\gamma}$, where W_γ are the weights of the Hamiltonians in the sum. This is achieved using the operator PREPARE(W), defined in equation (3).

(b) Create the superposition of states $|k\rangle$ encoded in unary, as given in equation (34), where the amplitudes correspond to the square roots of the weights for a truncated Taylor series. This is achieved using the controlled rotations by θ_k, depicted in figure 3, where the values of θ_k are given by equation (35). The overall operation performed in steps (a) and (b) is denoted PREPARE(β), and defined in equation (33).

(c) Use the ancillas prepared in steps (a) and (b) as controls for the operations V_j, defined in equation (30); this controlled operation is denoted SELECT(V), and defined in equation (32). This controlled operation corresponds to K controlled phase shifts and applications of SELECT(H), defined in equation (2). The result of SELECT(V) is that it applies a superposition of the terms in the truncated Taylor series expansion of the Hamiltonian evolution to the target state.

(d) Apply PREPARE$(\beta)^{\top}$ to invert the state preparation in steps (a) and (b). Then, if the ancilla qubits were measured as all zero, that would correspond to a success and give U_r applied to the target state.

(e) Apply oblivious amplitude amplification to obtain success with unit probability.

The gate count of the entire algorithm is thus r times the cost of implementing SELECT(V) plus the cost of implementing PREPARE(β). Though we implement SELECT(V) with $\mathcal{O}(NK)$ gates, our brute-force construction of PREPARE(W) led to a gate count for PREPARE(β) which scales as $\mathcal{O}(K\Gamma)$. Thus, the total gate count of our database algorithm scales as

$$\mathcal{O}(rK\Gamma) = \mathcal{O}\left(\frac{N^4 \Lambda t \log(Nt/\epsilon)}{\log\log(Nt/\epsilon)}\right) = \widetilde{\mathcal{O}}(N^8 t). \tag{46}$$

While this bound suggests an exponentially more precise algorithm than those based on Trotterization, in the remainder of our paper we discuss an even more efficient algorithm with improved dependence on N.

5. Evolution under integral Hamiltonians

In section 4 we analyzed the database algorithm for quantum simulating chemistry Hamiltonians in a manner that is exponentially more precise than Trotterization. The most costly part of that procedure is the implementation of PREPARE(W), as in equation (3), which prepares a superposition state with amplitudes that are given by integrals over spin–orbitals, as in equations (11) and (12). Instead of classically precomputing these integrals and implementing PREPARE(W) with a database, the strategy we introduce is to numerically sample the integrals on-the-fly using the quantum computer. Because of this, we call this the 'on-the-fly' algorithm. To accomplish this, we discretize the integrals as sums and design a circuit which returns the integrand of these integrals at particular domain points. The motivation for approximating integrals as sums comes from a direct analogy between the discretization of time in the Taylor series approach for simulating time-dependent Hamiltonians [34] and the discretization of space in Riemann integration.

In [34], the time-ordered exponential is approximated by a Taylor series up to order K, and the integrals are then discretized as follows on each segment

$$\begin{aligned}\mathcal{T}\exp\left[-\mathrm{i}\int_0^{t/r} H(t)\mathrm{d}t\right] &\approx \sum_{k=0}^{K}\frac{(-\mathrm{i})^k}{k!}\int_0^{t/r} \mathcal{T}H(t_k)\ldots H(t_1)\mathrm{d}\boldsymbol{t}\\ &\approx \sum_{k=0}^{K}\frac{(-\mathrm{i}t/r)^k}{\mu^k k!}\sum_{j_1,\ldots,j_k=0}^{\mu-1} H(t_{j_k})\ldots H(t_{j_1}),\end{aligned} \tag{47}$$

where $\mathcal{T}$ is the time ordering operator. Now let us suppose that our Hamiltonian does not change in time, but instead that the Hamiltonian itself is given as a definite integral over the spatial region $\mathcal{Z}$ so that

$$H = \int_{\mathcal{Z}} \mathcal{H}(\vec{z})\mathrm{d}\vec{z}. \tag{48}$$

The second quantized Hamiltonian given by equation (25) is similar to this, except it includes both terms h_{ij}, which are integrals over one spatial coordinate, and $h_{ijk\ell}$, which are integrals over two spatial coordinates. While those integrands are also defined over all space, the integrands decay exponentially in space so we can approximate them as definite integrals over the finite region $\mathcal{Z}$, having volume $\mathcal{V}$. Then we can approximate the integral by

$$H \approx \int_{\mathcal{Z}} \mathcal{H}(\vec{z})\mathrm{d}\vec{z} \approx \frac{\mathcal{V}}{\mu}\sum_{\rho=1}^{\mu}\mathcal{H}(\vec{z}_\rho), \tag{49}$$

where $\vec{z}_\rho$ is a point in the domain $\mathcal{Z}$ at the ρth grid point.

As in section 4, we begin by dividing t into r segments. We turn our attention to a single segment

$$\begin{aligned}U_r &\approx \exp\left[-\mathrm{i}\frac{t}{r}\int_{\mathcal{Z}}\mathcal{H}(\vec{z})\mathrm{d}\vec{z}\right]\\ &\approx \sum_{k=0}^{K}\frac{(-\mathrm{i}t/r)^k}{k!}\left(\int_{\mathcal{Z}}\mathcal{H}(\vec{z})\mathrm{d}\vec{z}\right)^k\\ &= \sum_{k=0}^{K}\frac{(-\mathrm{i}t/r)^k}{k!}\int_{\mathcal{Z}}\mathcal{H}(\vec{z}_1)\cdots\mathcal{H}(\vec{z}_k)\mathrm{d}\vec{\boldsymbol{z}},\end{aligned} \tag{50}$$

where in the second line we have performed a Taylor expansion of the propagator and truncated at order K. In equation (50), the bolded symbol $\vec{\boldsymbol{z}}$ indicates a vector of vectors. Like before, if $r \geqslant \|H\|t$ then the relationship between K and ϵ is given by equation (29). To approximate the integral, we divide it into μ regions of volume $\mathcal{V}/\mu$. We now have

$$U_r \approx \sum_{k=0}^{K} \frac{(-\mathrm{i}t\mathcal{V})^k}{r^k\mu^k k!}\left(\sum_{\rho=1}^{\mu}\mathcal{H}(\vec{z}_\rho)\right)^k = \sum_{k=0}^{K}\frac{(-\mathrm{i}t\mathcal{V})^k}{r^k\mu^k k!}\sum_{\rho_1,\ldots,\rho_k=1}^{\mu}\mathcal{H}(\vec{z}_{\rho_1})\cdots\mathcal{H}(\vec{z}_{\rho_k}). \tag{51}$$

For the second quantized Hamiltonian, the W_γ in equation (1) are integrals over scalar functions $w_\gamma(\vec{z})$ as in equation (4). Using this property it is clear that

$$\mathcal{H}(\vec{z}) = \sum_{\gamma=1}^{\Gamma} w_\gamma(\vec{z}) H_\gamma \tag{52}$$

and the segment U_r can be expressed as

$$U_r \approx \sum_{k=0}^{K}\frac{(-\mathrm{i}t\mathcal{V})^k}{r^k\mu^k k!}\sum_{\gamma_1,\ldots,\gamma_k=1}^{\Gamma}\sum_{\rho_1,\ldots,\rho_k=1}^{\mu} w_{\gamma_1}(\vec{z}_{\rho_1})\cdots w_{\gamma_k}(\vec{z}_{\rho_k})H_{\gamma_1}\cdots H_{\gamma_k}. \tag{53}$$

The second quantized Hamiltonian given in equation (25) is a sum of terms which are integrals over one spatial coordinate and terms which are integrals over two spatial coordinates. This case is easily accounted for by taking $\vec{z}$ to be a vector including both spatial coordinates, and $\mathcal{V}$ to be the product of the volumes for the two coordinates. One can take the terms with the integral over the single spatial coordinate to also be integrated over the second spatial coordinate, and divided by the volume of integration of that second coordinate to give the correct normalization. We may now proceed with the truncated Taylor series simulation as in section 4. Whereas our database algorithm required PREPARE(W) to create a superposition of states weighted by the W_γ, as in equation (3), our on-the-fly algorithm will need to create a superposition of states weighted by the scalar integrands $w_\gamma(\vec{z}_\rho)$.

As the first step, we discuss a method for dynamically decomposing each $w_\gamma(\vec{z})$ into a sum of terms which differ only by a sign, $w_{\gamma,m}(\vec{z})$. That is, the decomposition is of the form

$$w_\gamma(\vec{z}) \approx \zeta\sum_{m=1}^{M} w_{\gamma,m}(\vec{z}), \qquad w_{\gamma,m}(\vec{z}) \in \{-1, +1\}, \tag{54}$$

where ζ is the precision of the decomposition and

$$\zeta \in \Theta\left(\frac{\epsilon}{\Gamma\mathcal{V}t}\right), \qquad M \in \Theta(\max_{\vec{z},\gamma}\ |w_\gamma(\vec{z})|/\zeta). \tag{55}$$

The sum in equation (54) corresponds to $w_\gamma(\vec{z})$ rounded to the nearest multiple of 2 ζ, so

$$\left| w_\gamma(\vec{z}) - \zeta\sum_{m=1}^{M} w_{\gamma,m}(\vec{z}) \right| \leqslant \zeta. \tag{56}$$

To accomplish this on-the-fly, we perform logic on the output of SAMPLE(w) which acts as

$$\text{SAMPLE}(w)|\gamma\rangle|\rho\rangle|0\rangle^{\otimes\log M} = |\gamma\rangle|\rho\rangle|\widetilde{w}_\gamma(\vec{z}_\rho)\rangle, \tag{57}$$

where $\widetilde{w}_\gamma(\vec{z}_\rho)$ is the binary representation of $w_\gamma(\vec{z}_\rho)$ using $\log M$ qubits. In particular, we will need to determine whether the $w_{\gamma,m}(\vec{z})$ for a given value of m should be 1 or -1. Since the superposition we desire should be weighted by the square root of this coefficient, we need to prepare states that either do or do not have a phase factor $i = \sqrt{-1}$ so

$$|\ell\rangle|\widetilde{w}_\gamma(\vec{z}_\rho)\rangle \mapsto \begin{cases} |\ell\rangle|\widetilde{w}_\gamma(\vec{z}_\rho)\rangle & \widetilde{w}_\gamma(\vec{z}_\rho) > (2m-M)\zeta, \\ \mathrm{i}|\ell\rangle|\widetilde{w}_\gamma(\vec{z}_\rho)\rangle & \widetilde{w}_\gamma(\vec{z}_\rho) \leqslant (2m-M)\zeta, \end{cases} \tag{58}$$

where $|\ell\rangle = |\gamma\rangle|m\rangle|\rho\rangle$ and $|\widetilde{w}_\gamma(\vec{z}_\rho)\rangle$ was obtained from SAMPLE(w). The phase factor can be obtained using phase-kickback in the usual way. Then we apply SAMPLE(w) a second time to erase the register $|\widetilde{w}_\ell(\vec{z}_\rho)\rangle$. A single query to this circuit allows for the construction of PREPARE(w) with the same complexity as SAMPLE(w). Accordingly, the overall action of PREPARE(w) is

$$\text{PREPARE}(w)|0\rangle^{\otimes\log(L)} = \sqrt{\frac{1}{\lambda}}\sum_{\ell=1}^{L}\sqrt{\frac{\zeta\mathcal{V}}{\mu}w_{\gamma,m}(\vec{z}_\rho)}\,|\ell\rangle, \tag{59}$$

where $L \in \Theta(\Gamma M\mu)$ and

$$\lambda \equiv L\frac{\zeta\mathcal{V}}{\mu} \in \Theta(\Gamma\mathcal{V}\max_{\vec{z},\gamma}\ |w_\gamma(\vec{z})|). \tag{60}$$

Before explaining the integrand circuit we briefly comment on the additional resources required for the Taylor series simulation under a discretized, position-dependent, integrand Hamiltonian. As in the constant Hamiltonian case, we need one register with $\Theta(K)$ qubits to encode $|k\rangle$ and K registers of $\Theta(\log\Gamma)$ qubits to encode $|\gamma_1\rangle\cdots|\gamma_k\rangle$. However, we also need extra ancilla qubits to store the value of m, the grid point registers, as well as the value registers which are used by the integrand oracle SAMPLE(w). This represents an additional ancilla overhead of $\Theta(K\log(M\mu))$.

The sources of simulation error are also similar to the constant Hamiltonian case. As we show in section 6, we can approximate the integrals with discrete sums to precision ϵ at a cost that is logarithmic in $1/\epsilon$. The error due to the discrete sum is controlled by the choice of μ, which we need to select so that the resulting error per segment is less than ϵ/r. The most costly integrals (due to the size of their domain) will be the two-electron integrals in equation (12) which have integrands of the form

$$w_\gamma(\vec{z}) = h_{ijk\ell}(\vec{x}, \vec{y}) = \frac{\varphi_i^*(\vec{x})\varphi_j^*(\vec{y})\varphi_\ell(\vec{x})\varphi_k(\vec{y})}{|\vec{x} - \vec{y}|}, \tag{61}$$

where $\vec{x}$ and $\vec{y}$ represent the three spatial degrees of freedom of two separate electrons. In section 6, we bound the cost to the quantum algorithm of estimating the corresponding integrals.

To summarize, the on-the-fly algorithm is as described below.

(1) Decompose the Hamiltonian into an integral which is approximated as $\frac{\mathcal{V}}{\mu}\sum_{\rho=1}^{\mu}\mathcal{H}(\vec{z}_\rho)$, as in equation (49).

(2) Subdivide the simulation time t into $r = \lambda t/\ln(2)$ segments, where λ is defined in equation (60).

(3) Expand the evolution for time t/r by a truncated Taylor series as in equation (51). (Steps 1 to 3 are classical pre-processing.)

(4) For each segment perform the following steps.

  (a) Apply PREPARE(w), as defined in equation (59), to create a superposition of states $|\ell\rangle = |\gamma\rangle|m\rangle|\rho\rangle$ weighted by i or $-$i.

  (b) Create the superposition of states $|k\rangle$ encoded in unary, as given in equation (34), except with Λ replaced with λ.

  (c) Apply SELECT(V), i.e. K controlled phase shifts and applications of SELECT(H), to coherently execute all terms in the truncated Taylor series.

  (d) Apply the transpose of the state preparation in step (a) and step (b).

  (e) Apply oblivious amplitude amplification to obtain success with unit probability.

Note that the key difference between this algorithm and that described in section 4 is the state preparation in PREPARE(w), which corresponds to terms in the discretized integral. The superposition of unary-encoded states $|k\rangle$ is modified, but only in that Λ is replaced with λ. In the next section we detail how to construct the oracle for the discretized integral, and its cost.

6. The integrand oracle

In section 5, we showed how one can implement the truncated Taylor series simulation technique by replacing a superposition state having amplitudes given by integrals with a superposition state having amplitudes given by their integrands, as well as a way of decomposing those integrands. We begin this section by constructing a circuit which allows us to sample from the integrands as in equation (57). First, we will need a circuit which computes values of the N spin–orbital basis functions $\varphi_1(\vec{z}_\rho)$ to $\varphi_N(\vec{z}_\rho)$ at $\vec{z}_\rho$, a real-valued position vector at grid point ρ. The action of each these oracles is

$$Q_{\varphi_j}|\rho\rangle|0\rangle^{\otimes\log M} = |\rho\rangle|\widetilde{\varphi}_j(\vec{z}_\rho)\rangle, \tag{62}$$

where $\widetilde{\varphi}_j(\vec{z}_\rho)$ represents the binary expansion of $\varphi_j(\vec{z}_\rho)$ using $\log M$ qubits. We will need N different circuits of this form, one for each basis function $\varphi_1(\vec{z})$ to $\varphi_N(\vec{z})$. Usually, the molecular spin–orbital basis functions are represented as sums of Gaussians multiplied by polynomials [43]. In that case, the quantum circuit Q_{φ_j} can be implemented as a reversible classical circuit that evaluates and sums the Gaussians associated with $\varphi_j(\vec{z})$. For example, in the STO-nG basis set, each orbital is a linear combination of n Gaussian basis functions [43]. In general, Gaussian functions may be evaluated classically with complexity that is at most polylogarithmic in $1/\epsilon$ [45]. The use of Gaussians has a historical precedent; they are used because those functions are simple to integrate on a classical computer. However, the use of a Gaussian basis is not necessarily an optimal strategy for a quantum implementation. We leave open the problem of determining an optimal representation of molecular orbital basis functions for evaluation on a quantum computer and develop a strategy based on the model chemistries used in classical treatments of electronic structure.

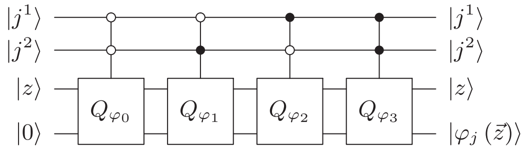

Figure 3. An oracle which returns the value of a particular basis function at a particular position $\vec{z}$ depending on the state of an ancilla register $|j\rangle$ which selects the oracle. Here j is represented in binary, where j^1 refers to the first bit of j, and j^2 refers to the second bit of j. This example is only valid when there are four basis functions.

Next, we combine N different Q_{φ_j} circuits, one for each $\varphi_j(\vec{z})$, to construct a circuit $\mathcal{Q}$ which allows us to apply any of the N basis functions. This circuit will have depth $\mathcal{O}(N\,\mathrm{polylog}(Nt/\epsilon))$ and may be constructed as the block diagonal operator

$$\mathcal{Q} = \prod_{j=1}^{N} |j\rangle\langle j| \otimes Q_{\varphi_j}. \tag{63}$$

Thus, $\mathcal{Q}$ is a sequence of Q_{φ_j} circuits with the spin–orbital selection completely controlled on a register encoding the basis function index j. There will be a factor of $\log(Nt/\epsilon)$ in the complexity because the controlled operations need to access $\mathcal{O}(\log N)$ qubits for j, as well as $\mathcal{O}(\log(Nt/\epsilon))$ qubits storing the position $\vec{z}$. In addition, the circuit needs to perform analytic operations (e.g. calculating exponentials for STO-nG), which will contribute an additional factor polynomial in $\log(Nt/\epsilon)$. An example implementation of $\mathcal{Q}$ for four basis functions is shown in figure 3.

We now discuss how one can use $\mathcal{Q}$ to compute the two-electron integrands in equation (61). To avoid singularities that would occur when two electrons occupy the same point in space, we change variables in equation (61) so that $\vec{\xi} = \vec{x} - \vec{y}$. With this substitution, the integral becomes

$$\int \frac{\varphi_i^*(\vec{x})\varphi_j^*(\vec{x}-\vec{\xi})\varphi_\ell(\vec{x})\varphi_k(\vec{x}-\vec{\xi})}{|\vec{\xi}|}\,\mathrm{d}^3\vec{\xi}\;\mathrm{d}^3\vec{x}. \tag{64}$$

Expressing $\vec{\xi}$ in spherical polar coordinates, with $\xi \equiv |\vec{\xi}|$, we have

$$\int \varphi_i^*(\vec{x})\varphi_j^*(\vec{x}-\vec{\xi})\varphi_\ell(\vec{x})\varphi_k(\vec{x}-\vec{\xi})\;\xi\sin\theta\;\mathrm{d}\xi\;\mathrm{d}\theta\;\mathrm{d}\phi\;\mathrm{d}^3\vec{x}. \tag{65}$$

We define the maximum value of any spin–orbital function as $\varphi_{\max}$ and the maximum value of its derivative in any direction as $\varphi'_{\max}$. In addition, we truncate the integral at a finite distance $x_{\max}$. Now assume that we discretize $\vec{x}$ in intervals of size δx along each degree of freedom. We can take the maximum value of ξ to be $x_{\max}$, and choose $\delta\xi = \delta x$, $\delta\theta = \delta\phi = \delta x / x_{\max}$.

The primary contribution to the complexity is in terms of the number of segments. The maximum value in the integrand of equation (65) is upper bounded by $x_{\max}\varphi_{\max}^4$. When discretizing the integral, each term in the sum is no larger than $\mathcal{O}(x_{\max}\varphi_{\max}^4\,\delta x^4(\delta x/x_{\max})^2) = \mathcal{O}(\varphi_{\max}^4\,\delta x^6/x_{\max})$ and there are $\mathcal{O}((x_{\max}/\delta x)^6)$ terms Multiplying these together gives us the contribution of the integral to the scaling of our on-the-fly algorithm,

$$\mathcal{O}(\varphi_{\max}^4\, x_{\max}^5), \tag{66}$$

which corresponds to the factor of $\mathcal{V}\max_{\vec{z},\gamma}|w_\gamma(\vec{z})|$ in equation (60). But how do $\varphi_{\max}$ and $x_{\max}$ scale with N? The maximum values $\varphi_{\max}$ are predetermined by the model chemistry, and hence are independent of N. Determining the appropriate value of $x_{\max}$ is a little more complicated.

Because the Hamiltonian is a sum of $\mathcal{O}(N^4)$ of the integrals, each integral should be approximated within error $\mathcal{O}(\epsilon/(N^4 t))$ to ensure that the final error is bounded by ϵ. Since the functions $\varphi_j(\vec{z})$ can be chosen to decay exponentially, $x_{\max}$ can be chosen logarithmically in the allowable error ϵ. The quantum chemistry problem is always defined within a particular basis, specified as part of a model chemistry [43]. The model chemistry prescribes how many spin–orbitals, how many basis functions, and what type of basis functions should be associated with each atom in a molecule. This includes a specification for parameters of the basis functions which impose a particular maximum value $\varphi_{\max}$, as well as a cutoff distance beyond which each $\varphi_j(\vec{z})$ is negligibly small. However, the space between basis functions on different atoms must grow as the cube root of N, because the molecular volume will grow as $\mathcal{O}(N)$. This would imply that the value of $x_{\max}$ needed scales as

$$x_{\max} \in \mathcal{O}(N^{1/3}\log(Nt/\epsilon)). \tag{67}$$

Nevertheless, each individual orbital $\varphi_j(\vec{z})$ is non-negligible on a region that grows only as $\mathcal{O}(\log N)$ for a given model chemistry. It is therefore advantageous to modify the grid used for the integral so it only includes points where one of the associated orbitals is non-negligible. This can be performed at unit cost if the center of each spin–orbital function is provided in an additional register when querying the circuit $\mathcal{Q}$. As above, the region where the orbital can be regarded as non-negligible can be chosen logarithmically in ϵ, to ensure that the overall error in the simulation is within error ϵ.

To be more specific, the coordinates for $\vec{x}$ can be chosen to be centered around the center of orbital φ_i, with the components of $\vec{x}$ only going up to a maximum value scaling as

$$x_{\max} \in \mathcal{O}(\log(Nt/\epsilon)). \tag{68}$$

For $\vec{\xi}$, we only wish to take values such that $\varphi_j(\vec{x} - \vec{\xi})$ are non-negligible. Here it should be noted that the spherical polar coordinates are only advantageous if we are in a region where $\vec{\xi}$ is near zero, where the Cartesian coordinates would have a divergence. In regions where $\vec{\xi}$ is large, the extra factor of ξ for the integral in spherical polar coordinates increases the complexity.

Therefore, if the minimum value of $|\vec{\xi}|$ such that $\varphi_j(\vec{x} - \vec{\xi})$ is non-negligible is $\mathcal{O}(\log(Nt/\epsilon))$, then the maximum value of $|\vec{\xi}|$ such that $\varphi_j(\vec{x} - \vec{\xi})$ is non-negligible will also be $\mathcal{O}(\log(Nt/\epsilon))$. Therefore we can use spherical polar coordinates, and obtain scaling as in equation (66) with $x_{\max}$ as in equation (68). On the other hand, if the minimum value of $|\vec{\xi}|$ such that $\varphi_j(\vec{x} - \vec{\xi})$ is non-negligible is $\Omega(\log(Nt/\epsilon))$, then we can use Cartesian coordinates, and the division by $|\vec{\xi}|$ can only lower the complexity. We obtain a contribution to the complexity scaling as $\mathcal{O}(\varphi_{\max}^4\, x_{\max}^3)$ with $x_{\max}$ as in equation (68). Here the power of $x_{\max}$ is 3 rather than 5, because we divide instead of multiplying by $|\vec{\xi}|$ as we did with spherical polar coordinates.

Next we consider the grid size needed to appropriately bound the error in the discretized integration. The analysis in the case where Cartesian coordinates are used is relatively straightforward. Considering a single block in six dimensions with sides of length δx, the value of the integrand can only vary by the maximum derivative of the integrand times δx (up to a constant factor). The error for the approximation of the integral on this cube is therefore that maximum derivative times δx^7. Then the number of these blocks in the integral is $\mathcal{O}((x_{\max}/\delta x)^6)$, giving an overall error scaling as $x_{\max}^6\, \delta x$ times the maximum derivative of the integrand.

The maximum derivative of the integrand can be bounded in the following way. For the derivative with respect to any component of $\vec{x}$, we obtain the derivative of the integrand scaling as

$$\mathcal{O}\left(\frac{\varphi'_{\max}\,\varphi_{\max}^3}{x_{\max}}\right), \tag{69}$$

where we have used the fact that we are only using Cartesian coordinates for $|\vec{\xi}| = \Omega(x_{\max})$. For the derivative of the integrand with respect to any component of $\vec{\xi}$ in the numerator of the integrand, the same scaling is obtained. For derivatives with respect to components of $\vec{\xi}$ in the denominator, the scaling is

$$\mathcal{O}\left(\frac{\varphi_{\max}^4}{x_{\max}^2}\right). \tag{70}$$

Overall, we therefore bound the error when discretizing in Cartesian coordinates as

$$\mathcal{O}((\varphi'_{\max} + \varphi_{\max}/x_{\max})\,\varphi_{\max}^3\, x_{\max}^5\, \delta x). \tag{71}$$

The analysis for spherical polar coordinates is a little more subtle, but it is largely equivalent if we scale the angular variables. It is convenient to define scaled angular variables

$$\theta' \equiv x_{\max}\theta, \qquad \phi' \equiv x_{\max}\phi. \tag{72}$$

Then the discretization lengths for all variables are δx. The volume of each block in the discretization is again δx^6, and there are $\mathcal{O}((x_{\max}/\delta x)^6)$ blocks. The total error is again therefore the maximum derivative of the integrand multiplied by $x_{\max}^6\, \delta x$.

The derivative of the integrand with respect to any component of $\vec{x}$ is again given by equation (69). Multiplication by ξ gives a factor $\mathcal{O}(x_{\max})$, but the change of variables to θ' and ϕ' gives division by a factor of $x_{\max}^2$. The derivative of the integrand with respect to ξ, θ' or ϕ' in any of the spin orbitals again gives a factor scaling as in equation (69). The derivative of the integrand with respect to ξ or θ' in $\xi\,\sin(\theta'/x_{\max})$ scales as in equation (70).

As a result, regardless of whether Cartesian coordinates are used or spherical polar coordinates, the error due to discretization is bounded as in equation (71). Thus, to achieve error in the integral no larger than $\mathcal{O}(\epsilon/(N^4 t))$, we require that

$$\delta x \in \Theta\left(\frac{\epsilon}{N^4 t}\frac{1}{(\varphi'_{\max} + \varphi_{\max}/x_{\max})\,\varphi_{\max}^3\, x_{\max}^5}\right). \tag{73}$$

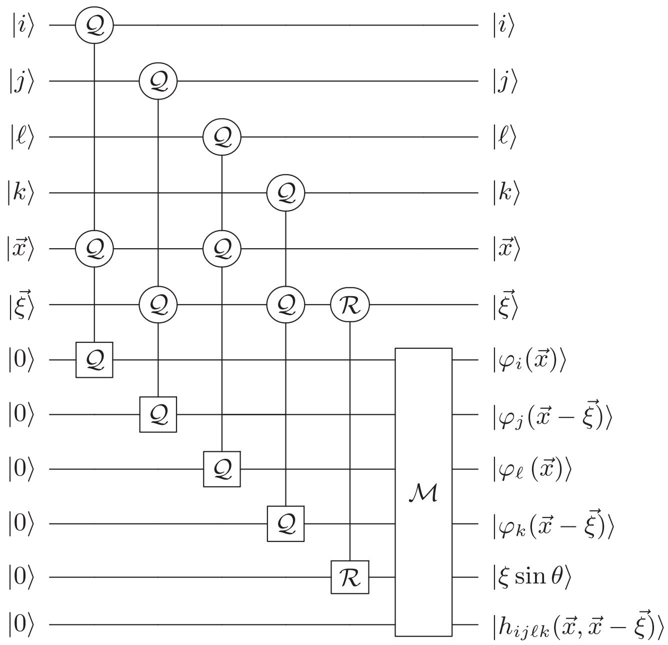
$|i\rangle$
$|j\rangle$
$|\ell\rangle$
$|k\rangle$
$|\vec{x}\rangle$
$|\vec{\xi}\rangle$
$|0\rangle$
$\mathcal{Q}$
$\mathcal{R}$
$\mathcal{M}$
$|\varphi_i(\vec{x})\rangle$
$|\varphi_j(\vec{x}-\vec{\xi})\rangle$
$|\varphi_\ell(\vec{x})\rangle$
$|\varphi_k(\vec{x}-\vec{\xi})\rangle$
$|\xi \sin\theta\rangle$
$|h_{ij\ell k}(\vec{x},\vec{x}-\vec{\xi})\rangle$

Figure 4. Circuit to sample the integrand of equation (65). The circuit combines four copies of $\mathcal{Q}$ with $\mathcal{R}$ and $\mathcal{M}$. The target registers for $\mathcal{Q}$ and $\mathcal{R}$ are denoted by boxes, and the control registers are denoted by circles.

The total number of terms in the sum then scales as

$$\mathcal{O}\left(\left(\frac{x_{\max}}{\delta x}\right)^6\right) = \Theta\left(\left(\frac{N^4 t}{\epsilon}(\varphi'_{\max} + \varphi_{\max}/x_{\max})\varphi^3_{\max}\, x^6_{\max}\right)^6\right). \tag{74}$$

This is quite large, but because we only need to use a number of qubits that is the logarithm of this, it only contributes a logarithmic factor to the complexity. Because the logarithm scales as $\mathcal{O}(\log(Nt/\epsilon))$, it contributes this factor to the complexity of SAMPLE(w).

Given $\mathcal{Q}$, computing the integrand in equation (65) is straightforward. We need to call $\mathcal{Q}$ four times on registers that contain $\vec{x}$ and $\vec{\xi}$. Let $\mathcal{R}$ denote a circuit computing the value of ξ sin θ when queried with the point $\vec{\xi}$. This circuit has the following action:

$$\mathcal{R}|\vec{\xi}\rangle|0\rangle = |\vec{\xi}\rangle|\xi \sin\theta\rangle. \tag{75}$$

The final element of our sampler circuit will be a reversible multiplier $\mathcal{M}$ which simply multiplies together five registers in a reversible fashion. This construction of SAMPLE(w) is shown in figure 4 and enables us to evaluate the integrand of equation (65), i.e.

$$\varphi_i^*(\vec{x})\varphi_j^*(\vec{x}-\vec{\xi})\varphi_\ell(\vec{x})\varphi_k(\vec{x}-\vec{\xi})\,\xi \sin\theta. \tag{76}$$

Next we consider how to construct a circuit for the one-electron integrals in equation (11). First, one constructs N additional circuits similar to the ones in equation (62) that return $\nabla^2\varphi_j(\vec{z})$ as opposed to $\varphi_j(\vec{z})$. These oracles are incorporated into a one-electron version of $\mathcal{Q}$ which is called along with a routine to compute the nuclear Coulomb interactions. The one-electron integrals have singularities at the positions of the nuclei. Similar to the two-electron integrals, these singularities can be avoided by using spherical polar coordinates. Each term in the sum over the nuclei should use spherical polar coordinates centered at that nucleus. Selection between the one-electron and two-electron routines is specified by $|\gamma\rangle$. Putting this together with the circuit in figure 4, we can implement SAMPLE(w) with $\mathcal{O}(N \log N)$ gates, and, as discussed in section 5, PREPARE(w) has the same complexity.

While the $\widetilde{\mathcal{O}}(N)$ gate count of PREPARE(w) is much less than the $\mathcal{O}(N^4)$ gate count of PREPARE(W), our on-the-fly algorithm requires more segments than the database algorithm. Whereas our database algorithm requires $r = \Lambda t/\ln(2)$ segments where Λ is the normalization in equation (3), our on-the-fly algorithm requires $r = \lambda t/\ln(2)$ segments where $\lambda \in \Theta(\Gamma\varphi^4_{\max}\, x^5_{\max})$ is the normalization in equation (59), which is accounted for in equations (60) and (66). Thus, by performing the algorithm in section 4 using PREPARE(w) instead of PREPARE(W) and taking $r = \lambda t/\ln(2)$, we see that our on-the-fly algorithm scales as

$$\widetilde{\mathcal{O}}(rNK) = \widetilde{\mathcal{O}}(NK\lambda t). \tag{77}$$

Using the scaling in equation (68), we can bound λ as

$$\lambda \in \mathcal{O}(\Gamma\varphi_{\max}^4 x_{\max}^5) \in \mathcal{O}(N^4[\log(Nt/\epsilon)]^5), \tag{78}$$

so that the overall gate count of the on-the-fly algorithm scales as

$$\widetilde{\mathcal{O}}(N^5 Kt) = \widetilde{\mathcal{O}}(N^5 t). \tag{79}$$

Recall that the $\widetilde{\mathcal{O}}$ notation indicates that logarithmic factors have been omitted. The full scaling includes a power of the logarithm of $1/\epsilon$.

To summarize, in this section we have provided the algorithm for the operation $\text{SAMPLE}(w)$ used in section 5. To achieve this operation, the key steps are:

(1) Convert from γ to (i, j, k, ℓ), and from the sampling point ρ to the corresponding values of $\vec{x}$ and $\vec{\xi}$.

(2) Apply the circuit shown in figure 4 to sample the integrand $h_{ijkl}(\vec{x}, \vec{x} - \vec{\xi}) = w_\gamma(\vec{z})$ (see equation (61)).

(3) The circuit of step 2 uses controlled-$\mathcal{Q}$ operations which calculate the value of an orbital $\varphi_j(\vec{z})$, and are performed using a circuit of the form in figure 3.

7. Discussion

We have introduced two novel algorithms for the simulation of molecular systems based primarily on the results of [34]. Our database algorithm involves using a database to store the molecular integrals; its gate count scales as $\widetilde{\mathcal{O}}(N^8 t)$. Our on-the-fly algorithm involves computing those integrals on-the-fly; its gate count scales as $\widetilde{\mathcal{O}}(N^5 t)$. Both represent an exponential improvement in precision over Trotter-based methods which scale as $\widetilde{\mathcal{O}}(N^9\sqrt{t^3/\epsilon})$ when using reasonably low-order decompositions, and over all other approaches to date.

Specifically, our database algorithm scales like $\widetilde{\mathcal{O}}(N^4\Lambda t)$ where we have used the bound $\Lambda \in \mathcal{O}(N^4)$. However, we believe this bound is very loose. As discussed in [8, 43], the use of local basis sets leads to a number of two-electron integrals that scales as $\widetilde{\mathcal{O}}(N^2)$ in certain limits of large molecules. Accordingly, the true scaling of the database algorithm is likely to be closer to $\widetilde{\mathcal{O}}(N^6 t)$. It also seems possible that our integration scheme is suboptimal; it is possible that it can be improved by taking account of smaller values of $h_{ijk\ell}$.

Our asymptotic analysis suggests that these algorithms will allow for the quantum simulation of molecular systems larger than would be possible using Trotter-based methods. However, numerical simulations will be crucial in order to further optimize these algorithms and better understand their scaling properties. Just as recent work showed significantly more efficient implementations of the original Trotterized quantum chemistry algorithm [5–9], we believe the implementations discussed here are far from optimal. Furthermore, just as was observed for Trotterized quantum chemistry [7, 9], we believe our simulations might scale much better when only trying to simulate ground states of real molecules. In light of this, numerical simulations may indicate that the scaling for real molecules is much less than our bounds predict.

Acknowledgments

The authors thank Jonathan Romero, Jarrod McClean, David Poulin, and Nathan Wiebe for helpful comments. DWB is funded by an Australian Research Council Future Fellowship (FT100100761) and a Discovery Project (DP160102426). PJL acknowledges the support of the National Science Foundation under grant PHY-0955518. AAG acknowledges the support of the Office of Naval Research under award: N00014-16-1-2008. AAG and PJL acknowledge the support of the Air Force Office of Scientific Research under award FA9550-12-1-0046.

References

[1] Barends R *et al* 2014 *Nature* **508** 500
[2] Kelly J *et al* 2015 *Nature* **519** 66
[3] Nigg D, Müller M, Martinez E A, Schindler P, Hennrich M, Monz T, Martin-Delgado M A and Blatt R 2014 *Science* **345** 302
[4] Córcoles A, Magesan E, Srinivasan S J, Cross A W, Steffen M, Gambetta J M and Chow J M 2015 *Nat. Commun.* **6** 6979
[5] Wecker D, Bauer B, Clark B K, Hastings M B and Troyer M 2014 *Phys. Rev.* A **90** 022305
[6] Hastings M B, Wecker D, Bauer B and Troyer M 2015 *Quantum Inf. Comput.* **15** 1
[7] Poulin D, Hastings M B, Wecker D, Wiebe N, Doherty A C and Troyer M 2015 *Quantum Inf. Comput.* **15** 361
[8] McClean J R, Babbush R, Love P J and Aspuru-Guzik A 2014 *J. Phys. Chem. Lett.* **5** 4368
[9] Babbush R, McClean J, Wecker D, Aspuru-Guzik A and Wiebe N 2015 *Phys. Rev.* A **91** 022311
[10] Berry D, Ahokas G, Cleve R and Sanders B 2007 *Commun. Math. Phys.* **270** 359

[11] Wiebe N, Berry D W, Hoyer P and Sanders B C 2011 *J. Phys. A Math. Theor.* **44** 445308
[12] Gibney E 2014 *Nature* **516** 24
[13] Mueck L 2015 *Nat. Chem.* **7** 361
[14] Lloyd S 1996 *Science* **273** 1073
[15] Abrams D S and Lloyd S 1997 *Phys. Rev. Lett.* **79** 2586
[16] Aspuru-Guzik A, Dutoi A D, Love P J and Head-Gordon M 2005 *Science* **309** 1704
[17] Whitfield J D, Biamonte J and Aspuru-Guzik A 2011 *Mol. Phys.* **109** 735
[18] Babbush R, Love P J and Aspuru-Guzik A 2014 *Sci. Rep.* **4** 6603
[19] Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love P J, Aspuru-Guzik A and O'Brien J L 2014 *Nat. Commun.* **5** 4213
[20] Yung M-H, Casanova J, Mezzacapo A, McClean J, Lamata L, Aspuru-Guzik A and Solano E 2014 *Sci. Rep.* **4** 3589
[21] McClean J R, Romero J, Babbush R and Aspuru-Guzik A 2016 *New J. Phys.* **18** 023023
[22] Cody Jones N, Whitfield J D, McMahon P L, Yung M-H, Meter R V, Aspuru-Guzik A and Yamamoto Y 2012 *New J. Phys.* **14** 115023
[23] Veis L and Pittner J 2010 *J. Chem. Phys.* **133** 194106
[24] Wang Y *et al* 2015 *ACS Nano* **9** 7769–74
[25] Whitfield J D 2013 *J. Chem. Phys.* **139** 021105
[26] Whitfield J D 2015 arXiv:1502.03771
[27] Li Z, Yung M-H, Chen H, Lu D, Whitfield J D, Peng X, Aspuru-Guzik A and Du J 2011 *Sci. Rep.* **1** 88
[28] Toloui B and Love P J 2013 arXiv:1312.2579
[29] Aharonov D and Ta-Shma A 2003 *Proc. 35th Annual ACM Symp. on Theory of Computing* (*STOC '03 ACM*) (*New York, NY, USA*) pp 20–9
[30] Berry D W, Ahokas G, Cleve R and Sanders B C 2007 *Commun. Math. Phys.* **270** 359
[31] Cleve R, Gottesman D, Mosca M, Somma R D and Yonge-Mallo D 2009 *Proc. 41st Annual ACM Symp. on Theory of Computing* (*STOC '09ACM*) (*New York, NY, USA*) pp 409–16
[32] Berry D W, Cleve R and Gharibian S 2014 *Quantum Inf. Comput.* **14** 1
[33] Berry D W, Childs A M, Cleve R, Kothari R and Somma R D 2014 *Proc. 46th Annual ACM Symp. on Theory of Computing* (*STOC '14 ACM*) (*New York, NY, USA*) pp 283–92
[34] Berry D W, Childs A M, Cleve R, Kothari R and Somma R D 2015 *Phys. Rev. Lett.* **114** 090502
[35] Jordan P and Wigner E 1928 *Z. Phys.* **47** 631
[36] Somma R D, Ortiz G, Gubernatis J, Knill E and Laflamme R 2002 *Phys. Rev.* A **65** 17
[37] Bravyi S and Kitaev A 2002 *Ann. Phys. NY* **298** 210
[38] Seeley J T, Richard M J and Love P J 2012 *J. Chem. Phys.* **137** 224109
[39] Tranter A, Sofia S, Seeley J, Kaicher M, McClean J, Babbush R, Coveney P V, Mintert F, Wilhelm F and Love P J 2015 *Int. J. Quantum Chem.* **115** 1431–41
[40] Abrams D S and Williams C P 1999 arXiv:quant-ph/9908083
[41] Grover L K 2000 *Phys. Rev. Lett.* **85** 1334
[42] Grover L K 1996 *Proc. 28th Annual ACM Symp. on Theory of Computing* (*STOC '96 ACM*) (*New York, NY, USA*) pp 212–9
[43] Helgaker T, Jorgensen P and Olsen J 2002 *Molecular Electronic Structure Theory* (New York: Wiley)
[44] Shende V, Bullock S and Markov I 2006 *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **25** 1000
[45] Brent R P and Zimmermann P 2010 *Modern Computer Arithmetic* (Cambridge: Cambridge University Press) p 236