Optimizing Electronic-Structure Hamiltonians for Locality

by

Louis Wenjun Marquis

S.B. Computer Science and Engineering, Mathematics
Massachusetts Institute of Technology, 2025

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2025

Authored by:	Louis Wenjun Marquis Department of Electrical Engineering and Computer Science May 9, 2025
Certified by:	Aram Harrow Professor of Physics, Thesis Supervisor
Accepted by:	Katrina LaCurts Department of Electrical Engineering and Computer Science Chair, Master of Engineering Thesis Committee

Optimizing Electronic-Structure Hamiltonians for Locality

by

Louis Wenjun Marquis

Submitted to the Department of Electrical Engineering and Computer Science
on May 9, 2025 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

The developments of the "kinetic theory" of gases made within the last ten years have enabled it to account satisfactorily for many of the laws of gases. The mathematical deductions of Clausius, Maxwell and others, based upon the hypothesis of a gas composed of molecules acting upon each other at impact like perfectly elastic spheres, have furnished expressions for the laws of its elasticity, viscosity, conductivity for heat, diffusive power and other properties. For some of these laws we have experimental data of value in testing the validity of these deductions and assumptions. Next to the elasticity, perhaps the phenomena of the viscosity of gases are best adapted to investigation.[1]

Thesis supervisor: Aram Harrow
Title: Professor of Physics

[1]Text from Holman (1876): doi:10.2307/25138434.

Acknowledgments

Write your acknowledgments here.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The field of computational chemistry aims to use computers to simulate and better understand the behavior of complex chemical systems in the real world. Drug discovery, materials science, and nanotechnology are only a few areas that would benefit from the ability to computationally predict behavior of molecular systems. However, chemical systems at the particle level are inherently shaped by quantum effects, which are not easily modeled by a classical (non-quantum) computer. In particular, the phenomenon of quantum superposition allows a system to be in a linear combination of an exponential number of basis states at the same time. A classical computer must numerically store this exponential amount of data, a prohibitive task.

Quantum computational chemistry aims to perform this task using quantum computers, which can represent such systems more easily because its hardware, by definition, is inherently quantum. A superposition in a chemical system, for example, can represented by physically putting a quantum computer's hardware into superposition.

1.1 Hamiltonian Simulation

In any quantum system, the behavior in which its state changes over time is dictated by its Hamiltonian operator $\hat{H}$. In particular, if the system starts in state $|\psi(0)\rangle$, then we can determine its state $|\psi(t)\rangle$ at time t.

$$|\psi(t)\rangle = e^{-i\hat{H}t} |\psi(0)\rangle \tag{1.1}$$

Operators and states can be thought of as matrices and vectors, but where a vector maps integer indices to values, a state is a function that maps a continuous variable (such as position) to an amplitude value. An operator maps state functions to state functions how a matrix maps vectors to vectors.

The Hamiltonian $\hat{H}$ depends on many factors, such as the nature of interactions between particles in the system or external forces such as electric or magnetic fields acting in the system. $\hat{H}$, and in turn $U = e^{-i\hat{H}t}$ can be very complex, and usually it's not feasible to simulate U exactly. Instead, we try find a different operator $\tilde{U}$ that is easy to simulate and sufficiently close to $\tilde{U}$. Usually we choose to define the error as the spectral norm (largest

singular value) of the difference between U and $\tilde{U}$. The goal is for the error to be below some variable threshold ϵ.

$$\epsilon \geq ||\tilde{U} - U|| = ||\tilde{U} - e^{-iHt}|| \tag{1.2}$$

This task is accordingly named Hamiltonian simulation.

One important Hamiltonian is the electronic structure Hamiltonian, describing the behavior of electrons in a molecule consisting of many nuclei and electrons. In modeling this system, we use the Born-Oppenheimer approximation, which assumes that the positions of the nuclei are essentially fixed. This is justified by the fact that the nuclei are far more massive than the electrons, so we can study the movement of electrons separately from the movement of the nuclei. With this assumption, the electronic structure Hamiltonian contains only terms for the electrons' kinetic energies, for the interactions between electrons and nuclei, and for the interactions among electrons.

$$\hat{H} = -\sum_p \frac{\hbar^2}{2m_e}\nabla_p^2 - \sum_{p,P} \frac{e^2}{4\pi\epsilon_0}\frac{Z_P}{|\vec{r}_p - \vec{R}_P|} + \frac{1}{2}\sum_{p\neq q} \frac{e^2}{4\pi\epsilon_0}\frac{1}{|\vec{r}_p - \vec{r}_q|} \tag{1.3}$$

$\vec{r}_p$ denotes the position of the pth electron, and $\vec{r}_P$ and Z_P denote the position and charge of the Pth nucleus.

The ultimate goal of electronic structure theory is to find the eigenvalues and eigenstates of $e^{-i\hat{H}t}$, or the states that are invariant (up to a phase) under $e^{-i\hat{H}t}$. The ability to simulate the behavior of $e^{-i\hat{H}t}$ on an arbitrary state can directly be used to find the eigenvalues using the quantum phase estimation algorithm. The details of this algorithm are not essential in the topic of this paper, the relevant fact is the power unlocked by performing $e^{-i\hat{H}t}$.

1.2 Trotterization

The Hamiltonian H of a system may often be a sum of many terms $\hat{h}_j$, as exemplified by the electronic structure Hamiltonian in (1.3).

$$\hat{H} = \sum_j \hat{h}_j \tag{1.4}$$

We can use the Lie-Trotter formula to approximate the target operator $U = e^{-i\hat{H}t}$ as the product of many smaller rotations.

$$U = e^{-i\hat{H}t} = e^{-i\sum_j \hat{h}_j t} = \lim_{r\to\infty} \prod_j e^{-i\hat{h}_j \frac{t}{r}} \tag{1.5}$$

In other words, for sufficiently large r, U is essentially equivalent to a series of much smaller evolutions with Hamiltonian $\hat{h}_j$. This is the first-order Trotter-Suzuki Algorithm. This is very convenient because it means that if we can simulate a Hamiltonian evolution with $\hat{h}_j$, we can also approximately simulate U. This will be the case most of the time in this paper.

However, it should be noted that r must be sufficiently large to make the approximation sufficiently precise. At the same time, the computation cost of the algorithm scales linearly

with the number of individual evolutions and therefore with r. It turns out that, to achieve an error threshold of ϵ, a cost of $\frac{L^3\Lambda^2t^2}{\epsilon}$ is necessary, in which L is the number of terms h_j, and $\Lambda = \max_j ||h_j||$.

First-order Trotterization as described is the most basic of such Hamiltonian simulation algorithms, and there is a rich field of literature finding improved methods to decrease error and computation cost for this problem.

One example is qDRIFT, a randomized version of first-order Trotterization. Its computation cost is $O(\frac{\lambda^2t^2}{\epsilon})$, in which $\lambda = \sum_j ||h_j||$.

The power of such Trotter-like techniques (breaking $e^{-i\hat{H}t}$ into many $e^{-i\hat{h}_jt}$) allows us to analyze each term $\hat{h}_j$ independently of the others. If one can simulate $e^{-i\hat{h}_jt}$ for every term, one can simulate U.

1.3 Second Quantized Fock States

We would like to model the electronic structure system described by $\hat{H}$ in (1.3). As previously described, the system consists of numerous identical electrons and numerous nuclei at fixed positions. The indistinguishability of the electrons means that the joint wavefunction is only determined by how many electrons are in each basis state, rather than which electron is in which state. For example, electrons 1 and 2 being in respective states ϕ_1 and ϕ_2 must create the same wavefunction as them being in the states ϕ_2 and ϕ_1. In addition, the joint wavefunction of many identical fermions must be anti-symmetric upon exchange of any pair of fermions. For example, if we want to express a state where one electron is in state ϕ_1 and the other is in state ϕ_2, a simple tensor product $\phi_1 \otimes \phi_2$ would not be a valid joint wavefunction. Instead, it must be anti-symmetrized.

$$\psi = \phi_1 \otimes \phi_2 - \phi_2 \otimes \phi_1 \tag{1.6}$$

This required process can make many-body calculations very cumbersome. Fortunately, second quantization significantly simplifies the representation of many-body systems by introducing the Fock basis, whose states are inherently anti-symmetrized.

Suppose for a single electron we'd like to use a basis $\{\phi_i\}$. Although the basis has infinite size, usually only basis states below a certain energy threshold are relevant, restricting our concern to a finite basis $\{\phi_i\}_{i\in[0,n)}$.

If there are many identical particles, we often want to express a state where f_i particles are in the ϕ_i state. We can label such a state with a vector $\vec{f}$. This is a Fock state $|\vec{f}\rangle$. For fermions (such as electrons), $f_i \in \{0, 1\}$ due to the Pauli-Exclusion Principle, so $\vec{f} \in \{0, 1\}^n$.

$$|\vec{f}\rangle = |f_0, f_1, ..., f_{n-1}\rangle \tag{1.7}$$

With n orbitals (basis states), there are 2^n such Fock states, one for each n-bitstring $\vec{f} \in \{0, 1\}^n$. These 2^n states form a complete basis of the many-body Hilbert space. That is, any state possible for a many-body system within these n orbitals can be expressed as a linear combination of these Fock states.

We can define fermionic creation $a_i^\dagger$ and annihilation a_i operators that map Fock states to Fock states.

$$\begin{aligned} a_i^\dagger |\vec{f}\rangle &= a_i^\dagger |f_0, ..., f_{n-1}\rangle \\ &= \delta_{f_i,1}(-1)^{\sum_{j=0}^{i-1}} |f_0, ..., f_i \oplus 1, ..., f_{n-1}\rangle \end{aligned} \tag{1.8}$$

$$\begin{aligned} a_i |\vec{f}\rangle &= a_i |f_0, ..., f_{n-1}\rangle \\ &= \delta_{f_i,0}(-1)^{\sum_{j=0}^{i-1}} |f_0, ..., f_i \oplus 1, ..., f_{n-1}\rangle \end{aligned} \tag{1.9}$$

$a_i^\dagger$ transforms a Fock state with $f_i = 0$ into a Fock state with $f_i = 1$, thereby "creating" a fermion in state ϕ_i and adding a phase shift to account for anti-symmetry. If orbital i is already occupied, the new state is 0 (it disappears). a_i does the reverse of $a_i^\dagger$ and de-occupies orbital i.

As typical with creation and annihilation operators, their product is a number operator $n_i = a_i^\dagger a_i$ [1].

$$n_i |\vec{f}\rangle = f_i |f_0, ..., f_{n-1}\rangle \tag{1.10}$$

The creation and annihilation operators have anti-commutation relations [1] that are analogous to typical commutation relations of creation and annihilation operators.

$$\{a_i^\dagger, a_j\} = \delta_{i,j} \tag{1.11}$$

$$\{a_i, a_j\} = \{a_i^\dagger, a_j^\dagger\} = 0 \tag{1.12}$$

Second quantization, in short, equips us with a concise representation of many-fermion wavefunction in the form of Fock states and operators $a_i^\dagger, a_i$ to easily manipulate them.

1.4 Jordan-Wigner Transformation

The Fock basis allows us to easily represent physical systems of many electrons. We would like to encode these physical systems on a quantum computer made of qubits so that we can compute on them. Conveniently, it is very straightforward to do so with the Fock basis.

Quantum computers are comprised of qubits. A qubit is simply a system with a two-dimensional Hilbert space. One example is an electron spin (up or down). Given a basis $\{|0\rangle, |1\rangle\}$, the state of a qubit can be any normalized linear combination of these two basis vectors.

$$|\psi\rangle = A_0 |0\rangle + A_1 |1\rangle \text{ such that } |A_0|^2 + |A_1|^2 = 1 \tag{1.13}$$

The joint state of n qubits can be any normalized superposition of all 2^n basis states $\{|\vec{x}\rangle\}_{\vec{x}\in\{0,1\}^n} = \{|x_0, x_1, ..., x_{n-1}\rangle\}_{\vec{x}\in\{0,1\}^n}$.

$$|\psi\rangle = \sum_{\vec{x}\in\{0,1\}^n} A_{\vec{x}} |\vec{x}\rangle \text{ such that } \sum_{\vec{x}\in\{0,1\}^n} |A_{\vec{x}}|^2 = 1 \tag{1.14}$$

It is then straightforward that basis states (1.7) of an n-orbital many-electron physical system correspond to basis states (1.14) of n qubits in a quantum computer. This is the Jordan-Wigner Encoding.

$$|\vec{f}\rangle \leftrightarrow |\vec{x}\rangle \tag{1.15}$$

That is, in the Jordan-Wigner Encoding, the value x_i of the ith qubit represents the occupation f_i of the orbital ϕ_i.

Accordingly, creation (1.8) and annihilation (1.9) operators on the physical system also correspond to qubit operators.

$$a_i \leftrightarrow Z_0 Z_1 ... Z_{i-1} \left|0\right\rangle \left\langle 1\right|_i \tag{1.16}$$

$$a_i^\dagger \leftrightarrow Z_0 Z_1 ... Z_{i-1} \left|1\right\rangle \left\langle 0\right|_i \tag{1.17}$$

$$n_i = a_i^\dagger a_i \leftrightarrow \left|1\right\rangle \left\langle 1\right|_i \tag{1.18}$$

We use the short-hand $U_i = I^{\otimes i} \otimes U \otimes I^{\otimes n-i-1}$ to denote applying a gate U on the ith qubit.

Observe that individual physical and annihilation operators map to a series of i gates (qubit operators). In the average case, this amounts to $O(n)$ gates, which is not ideal. There are other encodings that optimize the efficiency of operator mappings, but they sacrifice the simplicity in basis state mappings exhibited by the Jordan-Wigner Encoding. For the purposes of this project, the Jordan-Wigner Encoding is sufficient.

1.5 Double Factorization of Electronic Structure Hamiltonian

Equipped with Fock states and creation and annihilation operators, we can rewrite the electron structure Hamiltonian from (1.3) in a simpler, second-quantized form.

$$\hat{H} = \sum_{i,j} h_{ij} a_i^\dagger a_j + \frac{1}{2} \sum_{i,j,k,l} h_{ijkl} a_i^\dagger a_j^\dagger a_k a_l \tag{1.19}$$

The real coefficients h_{ij} and h_{ijkl} are the projections of $\hat{H}$ onto the chosen single-particle basis $\{\phi_i\}_{i\in[0,n)}$. These can be calculated on a classical computer.

$$\begin{aligned} h_{ij} &= \langle \phi_i | \left(-\frac{\hbar^2}{2m_e} \nabla^2 - \sum_I \frac{e^2}{4\pi\epsilon_0} \frac{Z_I}{|\vec{r} - \vec{R}_I|} \right) |\phi_j\rangle \\ &= \int \phi_i^*(\vec{r}) \left(-\frac{\hbar^2}{2m_e} \nabla^2 - \sum_I \frac{e^2}{4\pi\epsilon_0} \frac{Z_I}{|\vec{r} - \vec{R}_I|} \right) \phi_j(\vec{r}) d^3\vec{r} \end{aligned} \tag{1.20}$$

$$\begin{aligned} h_{ijkl} &= \langle \phi_i, \phi_j | \frac{e^2}{4\pi\epsilon_0} \frac{1}{|\vec{r}_1 - \vec{r}_2|} | \phi_l, \phi_k \rangle \\ &= \iint \phi_i^*(\vec{r}_1) \phi_j^*(\vec{r}_2) \frac{e^2}{4\pi\epsilon_0} \frac{1}{|\vec{r}_1 - \vec{r}_2|} \phi_l(\vec{r}_1) \phi_k(\vec{r}_2) d^3\vec{r}_1 d^3\vec{r}_2 \end{aligned} \tag{1.21}$$

Observe from (1.20) that h_{ij}, which is real, must be symmetric between i and j. From (1.21), h_{ijkl}, which is also real, must also be symmetric between i and l, j and k, and (i, l) and

(j,k). Inspired by this symmetry, we can rearrange the operators in the second summation using anti-commutation relations to put i,l and j,k next to each other.

$$\begin{aligned}
\hat{H} &= \sum_{i,j} h_{ij} a_i^\dagger a_j + \frac{1}{2} \sum_{i,j,k,l} h_{ijkl} a_i^\dagger a_j^\dagger a_k a_l \\
&= \sum_{i,j} h_{ij} a_i^\dagger a_j - \frac{1}{2} \sum_{i,j,k,l} h_{ijkl} a_i^\dagger a_j^\dagger a_l a_k \\
&= \sum_{i,j} h_{ij} a_i^\dagger a_j - \frac{1}{2} \sum_{i,j,k,l} h_{ijkl} a_i^\dagger (\delta_{jl} - a_l a_j^\dagger) a_k \\
&= \sum_{i,j} h_{ij} a_i^\dagger a_j - \frac{1}{2} \left(\sum_{i,j,k} h_{ijkj} a_i^\dagger a_k - \sum_{i,j,k,l} h_{ijkl} a_i^\dagger a_l a_j^\dagger a_k \right) \\
&= \sum_{i,j} (h_{ij} - \frac{1}{2} \sum_k h_{ikjk}) a_i^\dagger a_j + \frac{1}{2} \sum_{i,j,k,l} h_{ijkl} a_i^\dagger a_l a_j^\dagger a_k \\
&= \sum_{i,j} h'_{ij} a_i^\dagger a_j + \frac{1}{2} \sum_{i,j,k,l} h'_{iljk} a_i^\dagger a_l a_j^\dagger a_k \\
&= \hat{H}_1 + \hat{H}_2
\end{aligned} \tag{1.22}$$

We introduced new coefficients h'_{ij} and h'_{iljk} for this new ordering and label the summations $\hat{H}_1$ and $\hat{H}_2$.

$$\begin{aligned}
h'_{ij} &= h_{ij} - \frac{1}{2} \sum_k h_{ikjk} \\
\hat{H}_1 &= \sum_{i,j} h'_{ij} a_i^\dagger a_j \\
h'_{iljk} &= h_{ijkl} \\
\hat{H}_2 &= \frac{1}{2} \sum_{i,j,k,l} h'_{iljk} a_i^\dagger a_l a_j^\dagger a_k
\end{aligned} \tag{1.23}$$

As a reminder, we would like to simulate $U = e^{-i\hat{H}t}$ on a quantum computer. Trotterization allows us to study each term in (1.22) independently. That is, we just need to find a way to simulate $U_1 = e^{-i\hat{H}_1 t}$ and $U_2 = e^{-i\hat{H}_2 t}$ for arbitrary t. The direct brute force method is to substitute the operator correspondances from (1.16) and (1.17) into (1.22) to obtain the corresponding qubit Hamiltonian. With this method, it turns out that U_1 requires $O(n^2)$ gates and $U_2 = e^{-\hat{H}_2 t}$ requires $O(n^4)$ gates.

Recall that because $h'_{iljk} = h_{ijkl}$ is symmetric between (i,l) and (j,k), h' can be treated as a real symmetric $n^2 \times n^2$ matrix and eigendecomposed with eigenvalues λ_r and eigenvectors $Q^{(r)}$ whose indices are (i,l) ordered pairs. A Cholesky decomposition is also an option but for now we will use an eigendecomposition.

$$h'_{iljk} = \sum_r \lambda_r Q^{(r)}_{i,l} Q^{(r)}_{j,k} \tag{1.24}$$

$$\begin{aligned}
\hat{H}_2 &= \frac{1}{2}\sum_{i,j,k,l} h'_{ijkl} a_i^\dagger a_l a_j^\dagger a_k \\
&= \frac{1}{2}\sum_{i,j,k,l}\sum_r \lambda_r Q_{i,l}^{(r)} Q_{j,k}^{(r)} a_i^\dagger a_l a_j^\dagger a_k \\
&= \frac{1}{2}\sum_r \lambda_r \sum_{i,l} Q_{i,l}^{(r)} a_i^\dagger a_l \sum_{j,k} Q_{j,k}^{(r)} a_j^\dagger a_k \\
&= \frac{1}{2}\sum_r \lambda_r \left(\sum_{i,j} Q_{i,j}^{(r)} a_i^\dagger a_j\right)^2
\end{aligned} \tag{1.25}$$

Because h'_{iljk} is symmetric between i and l and between j and k, $Q_{i,j}^{(r)}$ is symmetric between i and j. Each n^2-length vector $Q^{(r)}$ can then be treated as an $n \times n$ matrix to be further eigendecomposed with eigenvalues $\lambda_s'^{(r)}$ and real orthonormal eigenvectors $R_s^{(r)}$ as the columns of $R^{(r)}$, which is orthogonal.

$$\begin{aligned}
Q^{(r)} &= R^{(r)} \begin{bmatrix} \lambda'_0 & & \\ & \ddots & \\ & & \lambda'_{n-1} \end{bmatrix} R^{(r)T} \\
Q_{i,j}^{(r)} &= \sum_s \lambda_s'^{(r)} R_{is}^{(r)} R_{js}^{(r)}
\end{aligned} \tag{1.26}$$

We introduce the operators $\tilde{a}_s^{(r)\dagger}, \tilde{a}_s^{(r)}$, which are respectively linear combinations of the the creation operators $\{a_i^\dagger\}_{i\in[0,n)}$ and annihilation operators $\{a_i\}_{i\in[0,n)}$. Because $R^{(r)}$ is orthogonal (and therefore unitary), these $\tilde{a}_s^{(r)\dagger}, \tilde{a}_s^{(r)}$ are actually creation and annihilation operators of a different Fock basis; that is, the basis $\{\tilde{\phi}_s = \sum_s R_{is}^{(r)} \phi_i\}_{s\in[0,n)}$. This rotated basis also has its own number operators $\tilde{n}_s^{(r)}$.

$$\begin{aligned}
\tilde{a}_s^{(r)\dagger} &= \sum_i R_{is}^{(r)} a_i^\dagger \\
\tilde{a}_s^{(r)} &= \sum_i R_{is}^{(r)} a_i \\
\tilde{n}_s^{(r)} &= \tilde{a}_s^{(r)\dagger} \tilde{a}_s^{(r)}
\end{aligned} \tag{1.27}$$

We can rewrite $\hat{H}_2$ in terms of $\tilde{a}_s^{(r)\dagger}, \tilde{a}_s^{(r)}, \tilde{n}_s^{(r)}$.

$$
\begin{aligned}
H_2 &= \frac{1}{2}\sum_r \lambda_r \left(\sum_{i,j} Q_{i,j}^{(r)} a_i^\dagger a_j\right)^2 \\
&= \frac{1}{2}\sum_r \lambda_r \left(\sum_{i,j}\sum_s \lambda_s'^{(r)} R_{is}^{(r)} R_{js}^{(r)} a_i^\dagger a_j\right)^2 \\
&= \frac{1}{2}\sum_r \lambda_r \left(\sum_s \lambda_s'^{(r)} \sum_i R_{is}^{(r)} a_i^\dagger \sum_j R_{js}^{(r)} a_j\right)^2 \\
&= \frac{1}{2}\sum_r \lambda_r \left(\sum_s \lambda_s'^{(r)} \tilde{a}_s^{(r)\dagger} \tilde{a}_s^{(r)}\right)^2 \\
&= \frac{1}{2}\sum_r \lambda_r \left(\sum_s \lambda_s'^{(r)} \tilde{n}_s^{(r)}\right)^2
\end{aligned}
\tag{1.28}
$$

We would like to simulate $U_2 = e^{-i\hat{H}_2}$; that is, implement efficiently its corresponding qubit operator. Since it's a summation over r, Trotterization again allows us to look at each term individually.

We would like to simulate $U_2^{(r)} = e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)}\tilde{n}_s^{(r)}\right)^2}$. If the operator's eigenbasis were the same as the qubits' computational basis, it would be very simple. However, its eigenbasis is the Fock basis constructed from the rotated basis $\{\tilde{\phi}_s\}$. Meanwhile, the qubit computational basis $\{|\vec{x}\rangle\}$ corresponds to the original physical Fock basis $\{|\vec{f}\rangle\}$ constructed from $\{\phi_i\}$. To deal with this, we have to essentially rotate the operator U_r into the computational basis. [2] uses the Thouless Theorem to show that such a rotation operator $U(R^{(r)})$ necessary can be calculated from $R^{(r)}$ and and corresponds to a maximum of $\binom{n}{2} = O(n^2)$ gates on the quantum computer.

$$
U_2^{(r)} = e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)}\tilde{n}_s^{(r)}\right)^2} = U(R^{(r)})^\dagger e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)} n_s\right)^2} U(R^{(r)}) = U(R^{(r)})^\dagger U_A^{(r)} U(R^{(r)})
\tag{1.29}
$$

We also need to find a qubit analog to $U_A^{(r)}$. One way is to expand the square $U_A^{(r)}$ and factor it into $O(n^2)$ individual operators, since the terms all commute.

$$
U_A^{(r)} = e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)} n_s\right)^2} = e^{-\frac{it}{2}\lambda_r \sum_{s,s'} \lambda_s'^{(r)}\lambda_{s'}'^{(r)} n_s n_{s'}} = \prod_{s,s'} e^{-\frac{it}{2}\lambda_r \lambda_s'^{(r)}\lambda_{s'}'^{(r)} n_s n_{s'}}
\tag{1.30}
$$

Each operator is the exponent of a product of two number operators $n_s, n_{s'}$. If $s = s'$, then that operator corresponds to a Z-rotation on a qubit. If $s \neq s'$, that operator corresponds to a controlled Z-rotation. In total this would comprise $O(n^2)$ gates.

Since $U_2^{(r)}$ can be implemented on the quantum computer with three steps each comprising $O(n^2)$ gates, it itself has gate complexity $O(n^2)$. However, the number of values of r is the rank of $\hat{H}_2$, which is bounded by n^2. Then one Trotter iteration of the terms in H_2 has gate

complexity $O(n^4)$, which does no better than the original brute force method, which had $O(n^4)$ terms each with constant gate count.

However, the $O(n^2)$ figure for the gate count of $U(R^{(r)})$ from [2] is only an upper bound, and much of the time $R^{(r)}$ is simple enough that $U(R^{(r)})$ can be implemented more cheaply, often in linear $O(n)$ gate count. The operator $U_A^{(r)}$ can also be implemented in sub-quadratic asymptotic gate count with a coherent method using quantum arithmetic circuits. This situation allows for a sub-quartic simulation of U_2, beating the naive brute force method.

This project attempted to explore the conditions for which a linear $O(n)$ gate count for $U(R^{(r)})$ arises, construct a quantum circuit design to implement the described coherent method, and analyze its cost and error. It further attempted some unconventional decomposition techniques on the electronic structure Hamiltonian that were inspired by the linear-gate count conditions, using gradient descent methods to optimize the decomposition.

Chapter 2

Quantum Arithmetic Circuit Design for Double-Factorized Electronic Structure Hamiltonian Simulation

We now propose and analyze a quantum circuit that approximately simulates $U_A^{(r)}$ from (1.30).

$$U_A^{(r)} = e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)} n_s\right)^2} \tag{2.1}$$

Observe that because $U_A^{(r)}$ is comprised of number operators n_s, its eigenbasis is simply the Fock basis $\{|\vec{f}\rangle\}_{\vec{f}}$. So when it acts on a Fock state $|\vec{f}\rangle$, its effect is simply a phase shift depending on the bits of $\vec{f}$.

$$U_A^{(r)}|\vec{f}\rangle = e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)} n_s\right)^2}|\vec{f}\rangle = e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)} f_s\right)^2}|\vec{f}\rangle \tag{2.2}$$

We can substitute the number operator correspondance from (1.18) to find the equivalent qubit operator.

$$U_A^{(r)} = e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)} n_s\right)^2} \leftrightarrow e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)} |1\rangle\langle 1|_s\right)^2} \tag{2.3}$$

As one may expect, this operator does the same thing on a qubit computational basis state $|\vec{x}\rangle$ that $U_A^{(r)}$ does on a Fock state, which is the phase rotation.

$$e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)} |1\rangle\langle 1|_s\right)^2}|\vec{x}\rangle = e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)} x_s\right)^2}|\vec{x}\rangle \tag{2.4}$$

We want to find an efficient way to approximately simulate $U_A^{(r)}$. That is, we want to find an operator $\tilde{U}_A^{(r)}$ that applies the specified phase rotation on $|\vec{x}\rangle$ as precisely as possible and with as few gates as possible. As a reminder, the error (and therefore precision) is measured by the operator norm of the difference in the ideal and approximate operators. ϵ is a parameter that limits the maximum error of this approximation and it affects the cost of $\tilde{U}_A^{(r)}$. A tighter bound requires more gates to achieve that bound.

$$\epsilon \geq ||\tilde{U}_A^{(r)} - U_A^{(r)}|| = ||\tilde{U} - e^{-\frac{it}{2}\lambda_r\left(\sum_s \lambda_s'^{(r)} n_s\right)^2}|| \tag{2.5}$$

(1.30) presented a method that simulated $U_A^{(r)}$ exactly (no error) but it required $O(n^2)$ gates. We now present a $\tilde{U}_A^{(r)}$ that requires $O(n(\log \frac{n}{\epsilon})^2)$ gates. The idea is to coherently calculate the phase of the phase rotation on a separate qubit register and use these qubits representing the phase as controls to apply a phase rotation.

More formally, denote the value of the summation in the phase rotation as $y(\vec{x})$.

$$\begin{aligned} y(\vec{x}) &= \sum_s \lambda_s'^{(r)} x_s \\ U_A^{(r)} |\vec{x}\rangle &= e^{-i\frac{t}{2}\lambda_r y_r(\vec{x})^2} |\vec{x}\rangle \end{aligned} \tag{2.6}$$

We begin with a register of n qubits in a Fock state $|\vec{x}\rangle$ and introduce an ancilla register of $\frac{m}{2}$ and another of m qubits, both initalized to zeros. We calculate an approximation $\tilde{y}_r(\vec{x})$ onto the smaller ancilla register. We use the value stored in the smaller register to compute an approximation $\tilde{y}_r(\vec{x})^2$ into the larger register. Then we use values of the m ancilla qubits to rotate the phase by $e^{-i\frac{t}{2}\lambda_r \tilde{y}_r(\vec{x})^2} |\vec{x}\rangle$. Finally, we uncompute the ancilla register and remove them afterwards. These steps comprise the proposed $\tilde{U}_A^{(r)}$.

$$|\vec{x}\rangle \to |\vec{x}\rangle |0^{\frac{m}{2}}\rangle |0^m\rangle \tag{2.7}$$
$$\to |\vec{x}\rangle |\tilde{y}_r(\vec{x})\rangle |0^m\rangle \tag{2.8}$$
$$\to |\vec{x}\rangle |\tilde{y}_r(\vec{x})\rangle |\tilde{y}_r(\vec{x})^2\rangle \tag{2.9}$$
$$\to e^{-i\frac{t}{2}\lambda_r \tilde{y}_r(\vec{x})^2} |\vec{x}\rangle |\tilde{y}_r(\vec{x})\rangle |\tilde{y}_r(\vec{x})^2\rangle \tag{2.10}$$
$$\to e^{-i\frac{t}{2}\lambda_r \tilde{y}_r(\vec{x})^2} |\vec{x}\rangle |0^{\frac{m}{2}}\rangle |0^m\rangle \tag{2.11}$$
$$\to e^{-i\frac{t}{2}\lambda_r \tilde{y}_r(\vec{x})^2} |\vec{x}\rangle \tag{2.12}$$
$$= \tilde{U}_A^{(r)} |\vec{x}\rangle \tag{2.13}$$

2.1 Summation

Because $\lambda'^{(r)}$ can assume any real (including irrational) values, a finite set of qubits will not be able to represent $y(\vec{x}) = \sum_s \lambda_s'^{(r)} x_s$ precisely. Instead, we compute an approximation $\tilde{y}_r(\vec{x})$ that can be expressed in $\frac{m}{2}$ bits. We essentially truncate (or round) each term in the sum and add them together. m can be increased to enhance precision at the cost of ancilla qubits and more gates, and vice versa. In short, we'd like to use the qubits $|\vec{x}\rangle$ to transform $|0^{\frac{m}{2}}\rangle$ to $|\tilde{y}_r(\vec{x})\rangle$.

$$|\vec{x}\rangle |0^{\frac{m}{2}}\rangle \to |\vec{x}\rangle |\tilde{y}_r(\vec{x})\rangle = |\vec{x}\rangle |\sum_s \tilde{\lambda}_s'^{(r)} x_s\rangle \tag{2.14}$$

The encoding between values z and qubit states must be able to handle negative numbers and must be able to hold the all possible values of $\tilde{y}_r(\vec{x})$. We define Y_r as the largest number

(by absolute value) that the register must hold, and we assign qubit j to represent a value $2^{M_r-(\frac{m}{2}-1)+j}$, where M_r is defined below. We use a signed method, so qubit $\frac{m}{2}-1$ represents the sign of the number.

$$Y_r = \max_{\vec{x}\in\{0,1\}^n} |y_r(\vec{x})| = \max\left(\sum_{s:\lambda_s^{(r)}>0} |\lambda_s^{(r)}|, \sum_{s:\lambda_s^{(r)}<0} |\lambda_s^{(r)}|\right) \tag{2.15}$$

$$M_r = \lfloor \log_2 Y_r \rfloor + 2 \tag{2.16}$$

$$|\vec{y}\rangle \leftrightarrow \tag{2.17}$$

Then this register can hold all values between -2^{M_r} and $2^{M_r} - 2^{M_r-(\frac{m}{2}-1)}$ that are multiples of $2^{M_r-(\frac{m}{2}-1)}$. By the definition of M_r, all possible $\tilde{y}_r(\vec{x})$ lie within this range. We can then define the bits $[\lambda_s^{(r)}]_j$ of each $\lambda_s^{(r)}$ in this signed binary form.

$$\lambda_s^{(r)} = \sum_{j=-\infty}^{\frac{m}{2}-1} (-1)^{\delta_{j,\frac{m}{2}-1}} [\lambda_s^r]_j 2^{M_r-(\frac{m}{2}-1)+j} \tag{2.18}$$

Since we only care about the bits for the values $2^{M_r-(\frac{m}{2}-1)}$ to 2^{M_r}, $\tilde{\lambda}_s^r$ is the accordingly truncated version of $\lambda_s^{(r)}$.

$$\tilde{\lambda}_s^r = \sum_{j=0}^{\frac{m}{2}-1} (-1)^{\delta_{j,\frac{m}{2}-1}} [\lambda_s^r]_j 2^{M_r-(\frac{m}{2}-1)+j} \tag{2.19}$$

As shown in A.1, (2.14) can be achieved by a series of controlled additions. There are many ways to implement such an adder, we chose the most direct way, which simply converts a classical adder into quantum circuits. A.2 shows how to implement a controlled addition of an arbitrary w in the described signed form.

$$w = \sum_{j=0}^{\frac{m}{2}-1} (-1)^{\delta_{j,\frac{m}{2}-1}} [w]_j 2^{M_r-(\frac{m}{2}-1)+j} \tag{2.20}$$

2.2 Squaring

We now have a register of $\frac{m}{2}$ qubits representing the value $\tilde{y}_r(\vec{x})$ and use this to calculate a value of $\tilde{y}_r(\vec{x})^2$ on the register of m qubits.

$$|\vec{x}\rangle\,|\tilde{y}_r(\vec{x})\rangle\,|0^m\rangle \to |\vec{x}\rangle\,|\tilde{y}_r(\vec{x})\rangle\,|\tilde{y}_r(\vec{x})^2\rangle \tag{2.21}$$

We now design a circuit that adds the square of an arbitrary signed $\frac{m}{2}$-qubit $|w\rangle$ to an m-qubit register.

$$|w\rangle\,|z\rangle \to |w\rangle\,|z+w^2\rangle \tag{2.22}$$

Because the smallest value represented by a qubit in the $|w\rangle$ register is $2^{M_r-(\frac{m}{2}-1)}$, the smallest value represented by a qubit in the $|z\rangle$ register will be $2^{2(M_r-(\frac{m}{2}-1))}$. So qubit k in the latter register will have value $(-1)^{\delta_{k,m-1}} 2^{2(M_r-(\frac{m}{2}-1))+k}$.

We can expand one of the w to turn w^2 into a sum.

$$w^2 = \sum_{j=0}^{\frac{m}{2}-1} (-1)^{\delta_{j,\frac{m}{2}-1}} [w]_j 2^{M_r-(\frac{m}{2}-1)+j} w \tag{2.23}$$

If one is to add w^2 to z, it's equivalent to adding each of these terms separately. The jth term in (2.23) is only nonzero if $[w]_j = 1$, so it's equivalent to a having a control on the jth qubit in $|w\rangle$. Adding $2^{M_r-(\frac{m}{2}-1)+j}w$ is equivalent to adding w but shifted up j qubits. Note that because $[w]_{\frac{m}{2}-1}$ is a sign bit, it must continue to be added to all bits in $|z\rangle$ until the end, accordingly with classical signed addition. A.3 shows a quantum circuit that adds $[w]_j 2^{M_r-(\frac{m}{2}-1)+j}w$. We can handle the $(-1)^{\delta_{j,\frac{m}{2}-1}}$, which flips the sign for $j = \frac{m}{2}-1$, by performing a controlled subtraction instead of an addition. A controlled subtraction is equivalent to the Hermitian (in this case, a reverse circuit) of the controlled addition.

Performing the circuit described in 2.23 using $w = \tilde{y}_r(\vec{x})$ will successfully accomplish 2.21.

2.3 Phase Rotation

We now have a register of m qubits in which the kth qubit has state $|[\tilde{y}_r(\vec{x})^2]_k\rangle$.

$$\tilde{y}_r(\vec{x})^2 = \sum_k (-1)^{\delta_{k,m-1}} [\tilde{y}_r(\vec{x})^2]_k 2^{2(M_r-(\frac{m}{2}-1))+k} \tag{2.24}$$

As described in (2.10), we'd like to apply a phase of $e^{-i\frac{t}{2}\lambda_r \tilde{y}_r(\vec{x})^2}$.

$$\begin{aligned}
|\vec{x}\rangle |\tilde{y}_r(\vec{x})\rangle |\tilde{y}_r(\vec{x})^2\rangle &\to e^{-i\frac{t}{2}\lambda_r \tilde{y}_r(\vec{x})^2} |\vec{x}\rangle |\tilde{y}_r(\vec{x})\rangle |\tilde{y}_r(\vec{x})^2\rangle \\
&= e^{-i\frac{t}{2}\lambda_r \sum_k (-1)^{\delta_{k,m-1}} [\tilde{y}_r(\vec{x})^2]_k 2^{2(M_r-(\frac{m}{2}-1))+k}} |\vec{x}\rangle |\tilde{y}_r(\vec{x})\rangle |\tilde{y}_r(\vec{x})^2\rangle \\
&= |\vec{x}\rangle |\tilde{y}_r(\vec{x})\rangle \prod_k e^{-i[\tilde{y}_r(\vec{x})^2]_k (-1)^{\delta_{k,m-1}} \frac{t}{2}\lambda_r 2^{2(M_r-(\frac{m}{2}-1))+k}} |\tilde{y}_r(\vec{x})^2\rangle
\end{aligned} \tag{2.25}$$

The individual phase rotation $e^{-i[\tilde{y}_r(\vec{x})^2]_k (-1)^{\delta_{k,m-1}} \frac{t}{2}\lambda_r 2^{2(M_r-(\frac{m}{2}-1))+k}}$ can be applied with a phase gate $P(\varphi)$ on qubit k with $\varphi = -(-1)^{\delta_{k,m-1}} \frac{t}{2}\lambda_r 2^{2(M_r-(\frac{m}{2}-1))+k}$. These gates are shown in (1.30).

$$P(\varphi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix} = |0\rangle\langle 0| + e^{i\varphi} |1\rangle\langle 1| \tag{2.26}$$

Once the phase has been applied, it is necessary to uncompute the values $\tilde{y}_r(\vec{x}), \tilde{y}_r(\vec{x})^2$ computed in the ancilla registers and reset the registers to zeros. One can simply do this by applying all the previous gates (in the summation and squaring) in reverse order. Then the empty registers can be safely removed and $\tilde{U}_A^{(r)}$ has successfully been applied.

2.4 Complexity Analysis

The controlled constant addition in A.2 requires $\frac{m}{2} - 1$ ancilla qubits (in addition to the target register) and its gate count depends on the values of $[w]_j$. At maximum, the gate count is $5.5m - 19$ Toffolis and $\frac{m}{2} - 1$ CNOTs. The Toffolis are far more expensive than CNOTs and there are far more of them, so the CNOTs can be ignored.

Next, we wish to establish an upper bound on the number of gates in the circuit in A.3. Observe that most of the gates have both a control on $|w\rangle_j$ and a control on a different qubit in $|w\rangle$, which is iterated. For some of these gates, the two controls will be the same qubit, and therefore is treated as one control. For simplicity of calculation, we will ignore that special case and treat it as two separate control qubits, since we only need an upper bound.

With this simplification, we observe that for each $|z\rangle_{j+2}, ..., |z\rangle_{m-1}$, there are 6 CCCNOTs and 2 CCNOTs (Toffolis), which is equivalent to 20 Toffolis. There are 2 additional Toffolis at the beginning and end. This is a total of $20(m - j - 2) + 2$ Toffolis in A.3.

Since the squaring protocol performs A.3 for each $j \in [0, \frac{m}{2})$, we can calculate the total number of Toffolis in the squaring protocol.

$$\begin{aligned}\sum_{j=0}^{\frac{m}{2}-1} (20(m - j - 2) + 2) &= 20m^2 - \frac{\frac{m}{2}(\frac{m}{2} - 1)}{2} - 40m + 2m \\ &= 17.5m^2 - 33m\end{aligned} \tag{2.27}$$

In our implementation of $\tilde{U}_A^{(r)}$, there are $2n$ controlled constant additions (n to compute and n to uncompute) and 2 squaring protocols (once to compute and uncompute). The phase rotations only contribute n single-qubit gates, which are negligible. We can calculate the total number of Toffolis.

$$2n(5.5m - 19) + 2(17.5m^2 - 33m) = 11mn - 38n + 35m^2 - 36m = O(mn + m^2) \tag{2.28}$$

2.5 Error Analysis

The size of the ancilla registers is determined by m and affects the precision of the applied phase. A larger m means more ancilla registers and a more precise phase, but quadratically increases the gate cost according to (2.28). If we want to guarantee an error below ϵ accordingly with (2.5), m must be sufficiently large. We now calculate the relation between ϵ and m.

2.6 Comparison

This is good when the number of Givens rotations is linear. When is that the case? I don't know.

Chapter 3

Convex Optimization of Decomposition of Electron-Electron Interaction Hamiltonian Term

3.1 Decomposition of Hamiltonian Term

3.2 Gradient Descent

3.3 Gate Count Analysis

3.4 Numerics

Chapter 4

Conclusion

4.1 Directions for Further Work

Appendix A

Quantum Circuit Diagrams

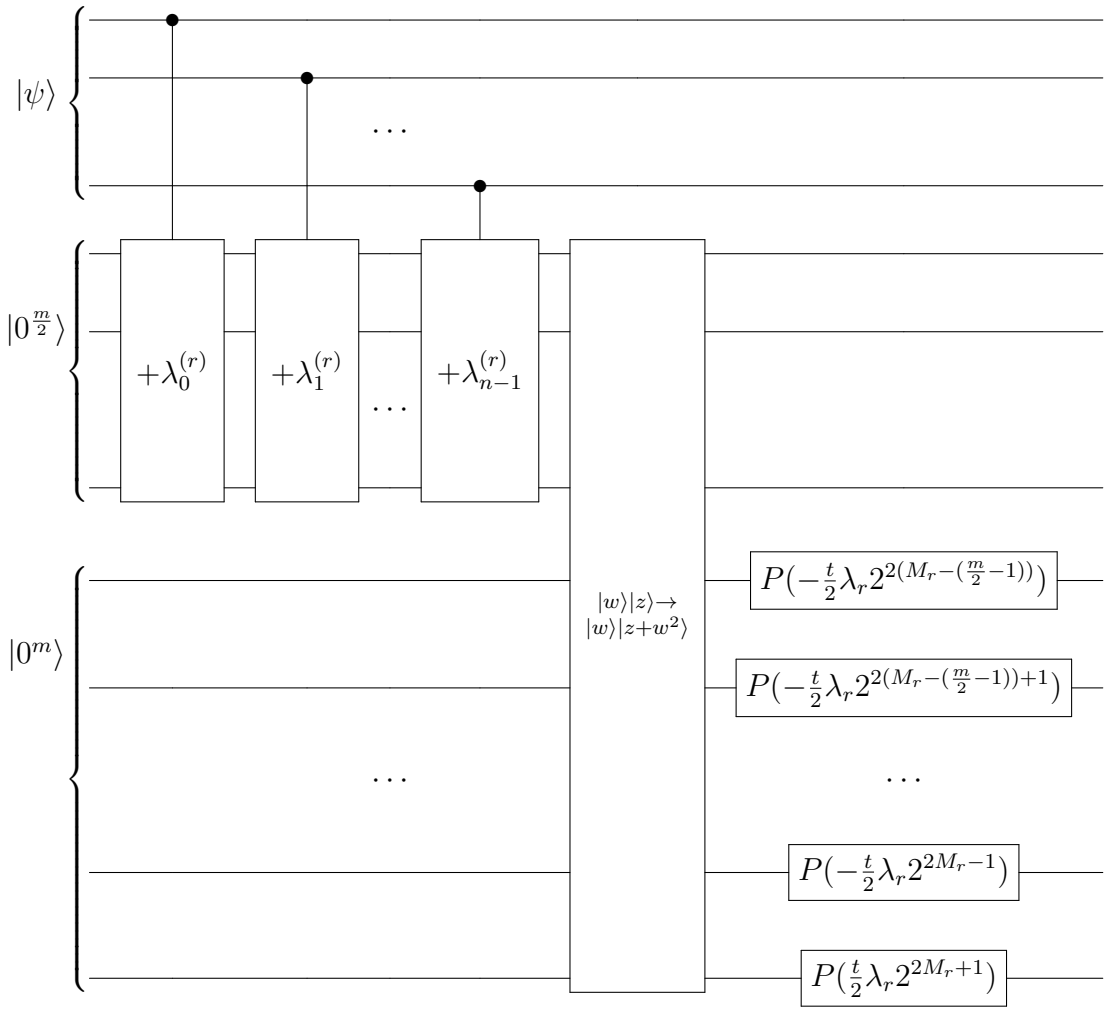

Figure A.1: Quantum circuit diagram for $U_A^{(r)}$ using arithmetic circuits (uncomputing not shown)

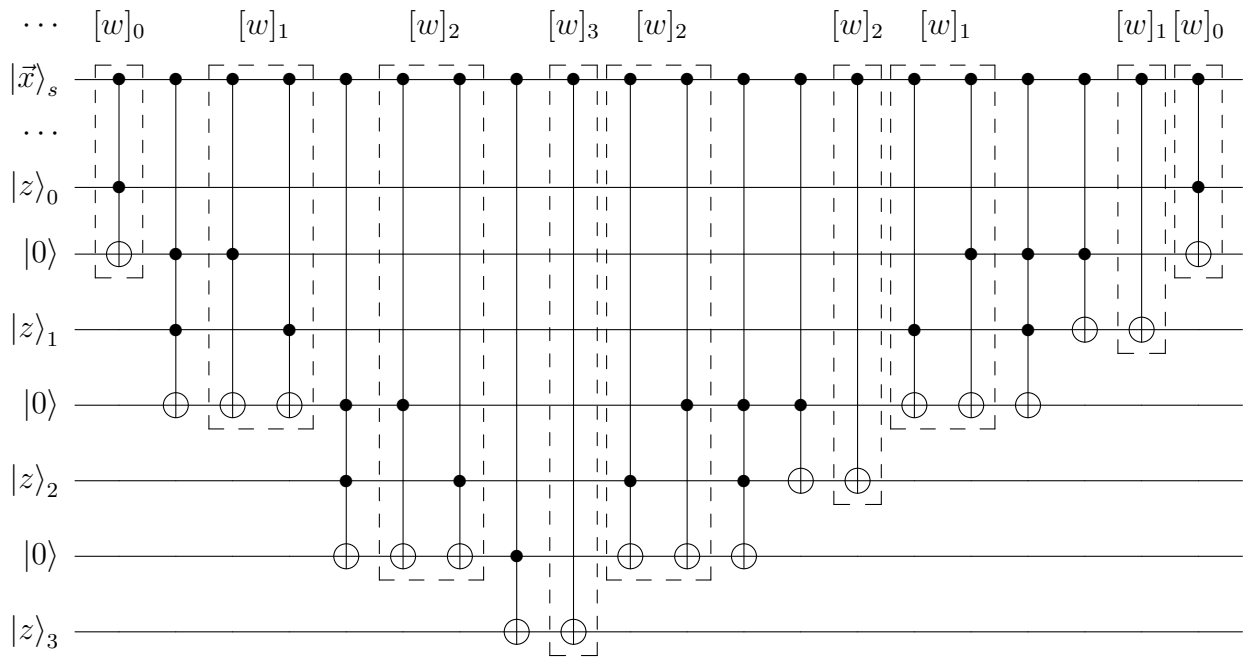

Figure A.2: Example quantum circuit diagram for a controlled addition of w for $\frac{m}{2} = 4$. The label $[w]_j$ indicates that the correspondingly boxed gates are only applied when $[w]_j = 1$.

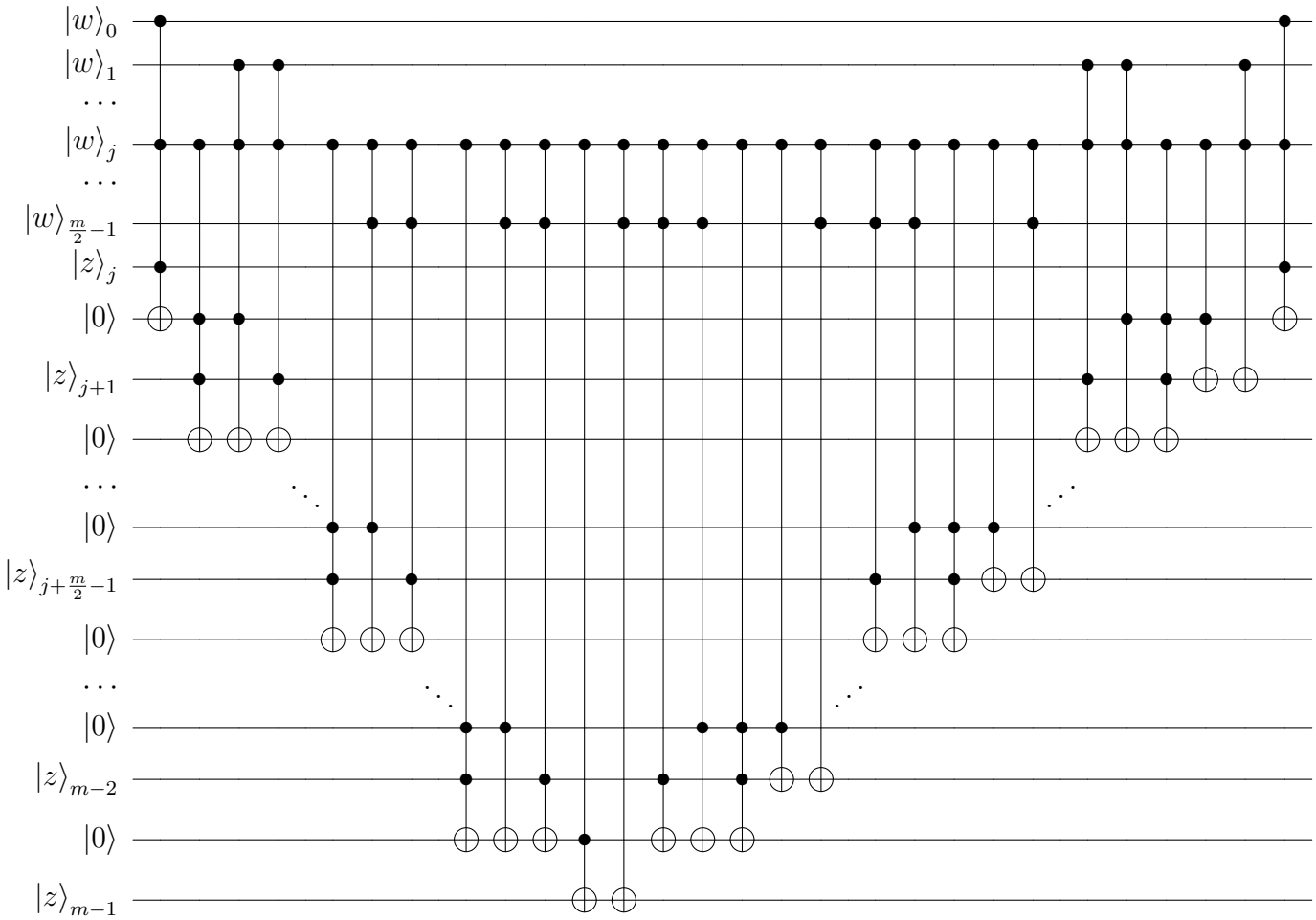

Figure A.3: Quantum circuit diagram for $|w\rangle|z\rangle \to |w\rangle|z + [w]_j 2^{M_r-(\frac{m}{2}-1)+j} w\rangle$. This is essentially the equivalent of A.2 if the added input were quantum, and if the controlling qubit $|\vec{x}\rangle_s$ were instead a qubit in w, and if the . Qubits 0 to $j-1$ of the $|z\rangle$ register are not shown.

References

[1] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan. "Quantum computational chemistry". In: *Reviews of Modern Physics* 92.1 (Mar. 2020). ISSN: 1539-0756. DOI: 10.1103/revmodphys.92.015003. URL: http://dx.doi.org/10.1103/RevModPhys.92.015003.

[2] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K.-L. Chan, and R. Babbush. "Quantum Simulation of Electronic Structure with Linear Depth and Connectivity". In: *Physical Review Letters* 120.11 (Mar. 2018). ISSN: 1079-7114. DOI: 10.1103/physrevlett.120.110501. URL: http://dx.doi.org/10.1103/PhysRevLett.120.110501.