Exact Gate Decompositions For Photonic Quantum Computers

Timjan Kalajdzievski

A Dissertation submitted to the Faculty of Graduate Studies in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Graduate program in Physics and Astronomy, York University, Toronto, Ontario

October, 2019

Abstract

The purpose of this work is to examine the use of decompositions on a continuous-variable quantum computer by both implementing and examining known methods, as well as to expand on them by developing my own. I detail the usage of known and new techniques for gate decompositions in some useful quantum algorithms such as simulating bosonic particles in a optical lattice, and solving differential equations with broad applications in other scientific fields. The new methods detailed in this work provide decompositions for continuous variable quantum computers which no longer require approximations. These methods rely on strategically using unitary conjugation and a lemma to the Baker-Campbell-Hausdorff formula to derive new exact decompositions from previously known ones, leading to exact decompositions for a large class of gates. I also demonstrate how exact decompositions can be employed in a wide range of algorithms, while requiring much fewer gates (sometimes as many as order-of-magnitude less) than equivalent decompositions with other methods. This work can potentially further bridge the gap between what is required to perform algorithms on a quantum computer and what can be done experimentally.

Acknowledgements

The author would like to thank his wife Liz, and parents Sasho and Nina for their support.

The author would also like to express gratitude toward long time mentor and supervisor Christian Weedbrook, as well as Juan Miguel Arrazola, Nathan Killoran, Tom Kirchner, Rene Fournier, and Marko Horbatsch for all of their help along the way.

Contents

List of Tables

List of Figures

1 Introduction

A practical quantum computer must be able to perform one or more quantum algorithms given to it. In order to do this, the operations of the algorithm must be translated into the logical operations directly implemented on the quantum computer. Theoretical methods to perform this translation have been extensively studied, but more so for some types of quantum computers than others [1–5]. The continued development of these methods has been the subject of my research leading up to this work.

I started my research in this subject with a publication looking into the simulation on a quantum computer of a Bose-Hubbard system of bosonic particles trapped in an optical lattice [6]. I then worked on my own method for implementing quantum algorithms in terms of the logical operations on a quantum computer [4], and demonstrated the use of my method in a quantum algorithm for solving partial differential equations [7]. This work is structured to follow closely my publications and starts with a general introduction to the topic in the rest of this Chapter, followed by a detailed overview of my method in Chapter 2. Chapter 3 then follows closely with my publication on simulating the Bose-Hubbard system using a mix of my method and others. Chapter 4 discusses two examples of algorithms for which my method is useful, the first of which is the quantum algorithm for solving partial differential equations detailed in [7]. Finally I provide a brief summary and discussion of the topic in Chapter 5.

To start, some definitions and overview on the implementation of quantum algorithms is needed. A quantum algorithm is an algorithm which is designed to run on a quantum computer. It is usually specified

by a sequence of high-level unitary transformations [6–11]. Unitary transformations are bounded linear operators that satisfy the following relation:

$$U^\dagger U = UU^\dagger = I, \tag{1}$$

where $U^\dagger$ is the adjoint of U, and I is the identity operator. A physical quantum computer, on the other hand, is only capable of performing a small set of elementary unitary operations which are referred to as gates. These physically implemented unitary operations, or gates, are the quantum computing analogue to the logical gates which are hardwired into a classical computer. The challenge of programming a quantum computer is to find combinations of elementary gates that can reproduce the operation of a desired quantum algorithm. It is known that specific sets of quantum logical gates exist such that any arbitrary unitary operation can be expressed as a finite product of gates from the set, to any desired precision [1, 2, 12–15]. Given their ability to reproduce any desired transformation, these are referred to as universal gates sets. Programming a quantum computer to perform a desired algorithm thus requires a method to decompose high-level unitaries in terms of universal gate sets. Ideally, a decomposition method will reproduce the algorithm with high precision while requiring as few gates as possible.

The qubit model of quantum computing uses two-state quantum systems called qubits as the basic units of information. Some physical examples include: vertical and horizontal polarized photons, or spin up and spin down electrons. The quantum gates acting on qubits are often represented as 2×2 unitary

matrices, and gate decompositions in this model of quantum computing have been well studied. For example, the Solovay-Kitaev theorem [5, 10] states that if a set of qubit gates generates a dense subset of $SU(2)$ (special unitary group of degree 2, which is the group of 2×2 unitary matrices with unit determinant), then it can approximate any $SU(2)$ unitary using a number of gates that is logarithmic in the precision. Stated informally, this means that any single-qubit operation can be approximated to high precision using a sequence of only a few gates. These results have been strengthened to even more efficient decompositions for single-qubit operations [16–19] and general multi-qubit operations [11].

In the continuous-variable (CV) model of quantum computing, registers are infinite-dimensional quantum systems – namely quantum harmonic oscillators – and the logic gates are unitaries acting on the infinite-dimensional Hilbert space [6–9, 14, 20–22]. This presents unique challenges for the task of decomposing arbitrary operations in CV photonic quantum computers, where comparatively less progress has been made thus far. Ref. [14] introduced the notion of universality in CV quantum systems based on the commutator algebra of quadrature operators. Following this, Ref. [3] presented the first systematic approach for decomposing arbitrary CV transformations, while Refs. [6–9] deal with decompositions for specific tasks. All these methods are approximate in the sense that the resulting sequence of gates from the universal set only implements the desired unitary up to a certain error, which can be decreased arbitrarily by employing longer circuits [3, 23, 24]. However, this can lead to very large circuit depths even if a modest precision is desired. Exact decompositions are known for a few specific cases [3, 7, 8], but it is not well understood what transformations allow exact decompositions nor how they can be derived.

1.1 Continuous-Variable Quantum Computing

In the CV model of quantum computing, each register is a quantum harmonic oscillator with corresponding creation and annihilation operators $\hat{a}_j^\dagger$ and $\hat{a}_j$, where the subscript refers to the mode they act upon. For definiteness, we henceforth assume that these registers are modes of the quantized electromagnetic field. The annihilation and creation operators satisfy the bosonic commutation relations $[\hat{a}_j, \hat{a}_k^\dagger] = \delta_{jk}$, and $[\hat{a}_j, \hat{a}_k] = [\hat{a}_j^\dagger, \hat{a}_k^\dagger] = 0$. An equivalent operator description of a bosonic system uses the quadrature field operators $\hat{x}$ and $\hat{p}$, which are related to the annihilation and creation operators as

$$\hat{x}_j = \frac{1}{2}\left(\hat{a}_j^\dagger + \hat{a}_j\right), \tag{2}$$

$$\hat{p}_j = \frac{i}{2}\left(\hat{a}_j^\dagger - \hat{a}_j\right), \tag{3}$$

with commutator $[\hat{x}_j, \hat{p}_j] = \frac{i}{2}$. This representation is equivalent to the choice of $\hbar = \frac{1}{2}$. The quadrature field operators represent dimensionless observables and are analogous to the position and momentum operators of a quantum harmonic oscillator. They each have corresponding eigenstates

$$\hat{x}\,|x\rangle = x\,|x\rangle\,,\ \hat{p}\,|p\rangle = p\,|p\rangle\,, \tag{4}$$

with real and continuous eigenvalues x and p. It is important to note that the eigenstates of $\hat{x}$ and $\hat{p}$ form two basis sets which are Fourier transforms of one another. On an arbitrary wavefunction in position space the quadrature operators have the following action

$$\hat{x}_k \left|f\right\rangle = \hat{x}_k \int dx^n f(\mathbf{x}) \left|\mathbf{x}\right\rangle = \int dx^n x_k f(\mathbf{x}) \left|\mathbf{x}\right\rangle \tag{5}$$

$$\hat{p}_k \left|f\right\rangle = \hat{p}_k \int dx^n f(\mathbf{x}) \left|\mathbf{x}\right\rangle = \frac{-i}{2} \int dx^n \frac{\partial}{\partial x_k} f(\mathbf{x}) \left|\mathbf{x}\right\rangle, \tag{6}$$

for all $k = 1, \ldots, n$. Note that the action of the momentum operator is equivalent to differentiation with respect to position when acting on a position state.

Quantum computing using quadrature operators to form some logic gates was first proposed by Seth Lloyd and Samuel L. Braunstein [14]. These logic gates were formed using Hamiltonians which were polynomial in the quadrature operators. For example, applying the Hamiltonian $\hat{x}_1\hat{x}_2\hat{p}_3$ will add the product of $\hat{x}_1\hat{x}_2$ to the $\hat{x}_3$ register. To form more arbitrary logic gates and examine their optical implementation, operations can be expressed as an exponential of quadrature operators.

1.2 Gate Decomposition

A universal gate set is a collection of gates such that any arbitrary unitary operation can be expressed as a finite series of gates from the universal set, to any chosen approximation. We focus on the universal

set specified by the gates

$$\{e^{i\frac{\pi}{2}(\hat{x}_j^2+\hat{p}_j^2)},\, e^{it_1\hat{x}_j},\, e^{it_2\hat{x}_j^2},\, e^{it_3\hat{x}_j^3},\, e^{i\tau\hat{x}_j\hat{x}_k}\}, \tag{7}$$

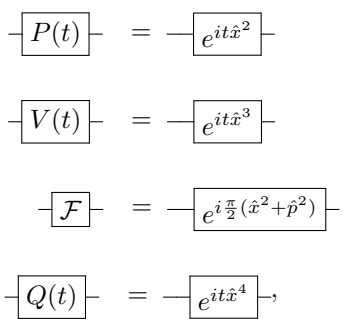
where t_1, t_2, t_3, and τ are real parameters. This particular universal set is chosen for mathematical convenience in our method. Four of the gates in the set have powers in the quadrature operators which are two or less. These, as well as operations of similar order, are referred to as Gaussian. The gate $e^{it_3\hat{x}_j^3}$ is the only non-Gaussian element in the universal set, the gate $e^{i\tau\hat{x}_1\hat{x}_2}$ allows for decompositions of multiple modes, while the Fourier transform gate $\mathcal{F} = e^{i\frac{\pi}{2}(\hat{x}^2+\hat{p}^2)}$ has the effect of mapping between the quadrature operators:

$$\mathcal{F}^\dagger\hat{x}\mathcal{F} = -\hat{p}, \tag{8}$$

$$\mathcal{F}^\dagger\hat{p}\mathcal{F} = \hat{x}, \tag{9}$$

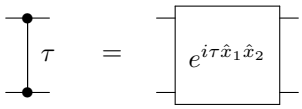
where these mappings follow directly from a lemma to the Baker-Campbell-Hausdorff formula, given in Eq. (28).

To simplify circuits used in later sections we can introduce the following notation:



(10)

where the final gate is refered to as the quartic gate and is not included in the universal set, but will be decomposed in terms of gates that are in a later section. The two-mode Cz or C-PHASE gate is given by



(11)

with tunable strength parameter τ.

An example of an equivalent universal set is one where the choice of non-Gaussian gate $e^{it_3\hat{x}^3}$, is replaced by the Kerr gate $e^{it_3\left(\hat{x}^2+\hat{p}^2\right)^2}$. In fact it is possible to replace the non-Gaussian gate with any other and retain universality, as well as removing one of the chosen Gaussian gates by showing that it can be expressed as a decomposition of another in the set [3]. A possible concern in either of these cases is whether the gates in the chosen set are costly to implement experimentally. It may be beneficial to keep a gate included in the set if its decomposition includes multiples of another, harder to implement gate.

For example the gate $e^{it_2\hat{x}^2}$ can be expressed in terms of the gate $e^{it_3\hat{x}^3}$ which eliminates the need for it in the set. But as the gate $e^{it_3\hat{x}^3}$ is much more complex to implement experimentally, it is commonplace to still include $e^{it_2\hat{x}^2}$.

For convenience, we express an arbitrary unitary as $U = e^{it\hat{H}}$ with $\hat{H} = \sum_{j=1}^{N} \hat{H}_j$ a Hermitian operator. When decomposing gates into a universal set, it is often necessary to express this sum of operators in the exponent as a product of exponential operators. More specifically, for $\hat{H} = \hat{A} + \hat{B}$ where $\hat{A}$ and $\hat{B}$ are Hermitian operators, the Zassenhaus formula [25] states that

$$e^{it(\hat{A}+\hat{B})} = e^{it\hat{A}} e^{it\hat{B}} e^{\frac{t^2}{2}[\hat{A},\hat{B}]} e^{\frac{-it^3}{6}\left(2[\hat{B},[\hat{A},\hat{B}]]+[\hat{A},[\hat{A},\hat{B}]]\right)} \dots \tag{12}$$

In the trivial case where $[\hat{A}, \hat{B}] = 0$ the product ends immediately after the first two operations. In general, however, it is possible that this product never terminates, resulting in a decomposition that is no longer finite. In this case, it is possible to truncate the product at a designated stage in the expansion and neglect the remaining commutators. This strategy is referred to as a Trotter-Suzuki approximation [26], which can be stated in the general case as

$$e^{it\hat{H}} = \left(\prod_{j=1}^{N} e^{i\frac{t}{K}\hat{H}_j} \right)^K + O(t^2/K), \tag{13}$$

where $\hat{H} = \sum_{j=1}^{N} \hat{H}_j$. This approximation requires $K = O(1/\varepsilon)$ gates to achieve precision ε for fixed t.

Note that the subscript j on $\hat{H}_j$ is only an index and does not refer to a mode, as each $\hat{H}_j$ may contain any number of modes.

In general, the unitaries of the form $e^{it\hat{H}_j}$ are not part of the universal set, so the task remains to decompose them. One way to achieve this is via the commutator approximation method detailed in Ref. [3]. This technique expresses sums and products of the quadrature operators in terms of commutators and then approximates the exponentials of these commutators as repeated products of their arguments. More specifically, given two Hermitian operators $\hat{A}$ and $\hat{B}$, it holds that [27]

$$e^{t^2[\hat{A},\hat{B}]} = \left(e^{i\frac{t}{K}\hat{B}}e^{i\frac{t}{K}\hat{A}}e^{-i\frac{t}{K}\hat{B}}e^{-i\frac{t}{K}\hat{A}}\right)^{K^2} + O(t^4/K). \tag{14}$$

For fixed t, $K = O(1/\varepsilon)$ gates are required to achieve an error of ε in the approximation, but the resulting circuit will have a depth of $O(1/\varepsilon^2)$. This means that very large circuits are required for even a modest precision. To illustrate the use of the commutator approximation technique, consider an example where we wish to decompose the operator $e^{it\left(\hat{x}^2\hat{p}+\hat{p}\hat{x}^2\right)}$. First, using the equality $\hat{x}^2\hat{p} + \hat{p}\hat{x}^2 = \frac{2}{3}[\hat{x}^3, \hat{p}^2]$ from Ref. [3], we have

$$e^{it\left(\hat{x}^2\hat{p}+\hat{p}\hat{x}^2\right)} = e^{\frac{2t}{3}[\hat{x}^3,\hat{p}^2]}. \tag{15}$$

Using Eq. (14) with $\hat{A} = \hat{x}^3$ and $\hat{B} = \hat{p}^2$ leads to

$$e^{\frac{2t}{3}[\hat{x}^3,\hat{p}^2]} = \left(e^{i\frac{\sqrt{\frac{2t}{3}}}{K}\hat{p}^2} e^{i\frac{\sqrt{\frac{2t}{3}}}{K}\hat{x}^3} e^{-i\frac{\sqrt{\frac{2t}{3}}}{K}\hat{p}^2} e^{-i\frac{\sqrt{\frac{2t}{3}}}{K}\hat{x}^3} \right)^{K^2} + O\left[\left(\frac{2t}{3}\right)^2 / K \right]. \tag{16}$$

Each of the gates on the right-hand side are contained within the universal set up to Fourier transforms, but in order to obtain a precision of $O(1/K)$, the product must be repeated $O(K^2)$ times. For instance, for $t = 1$, if the goal is to impose a precision of 10^{-3}, the product of four gates needs to be repeated approximately 10^5 times.

In fact, Ref. [28] examines the experimental error of implementing a sequence of gates on a qubit quantum computer. The results show that as the number of gates is increased, the accumulated physical implementation error eventually supersedes the precision gain from the repetitions. Thus, at some point, more repetitions do not lead to lower errors. This problem remains on a CV quantum computer and further study is required to determine the optimal trade-off between physical error in implementation and precision error in the decomposition. However, if it is possible to find an exact decomposition, then there is no longer any need for this trade-off.

In the literature on CV decompositions there are specific examples where the commutator approximation and even sometimes Trotter-Suzuki can be bypassed [3, 7, 8]. These cases are desirable, but no general framework has been proposed to characterize the set of gates admitting exact decompositions.

1.3 Optical Implementation

The most basic operations on a photonic quantum computer can be implemented directly by using linear optics, whereas higher-order operations are more complex and contain probabilistic elements. The universal set as in Eq. (7) has five elements, four of which have powers in the quadrature operators which are less then three. These gates, as well as other gates of similar order, can all be represented as combinations of optical rotations (or phase shifts), displacements, either squeezing or shearing, and if acting on multiple modes then also beamsplitters [21]. These optical elements are given by the following:

$$R(\theta) = e^{i\theta(\hat{x}^2+\hat{p}^2)/2}, \tag{17}$$

rotates a state in phase space by θ. When $\theta = \pi$ we have the Fourier transform gate from the universal set.

$$Z(s) = e^{is\hat{x}}, \tag{18}$$

a quadrature displacement of $\frac{s}{2}$ in momentum. When $s = t_1$ we have the second gate in our universal set. Note that a quadrature displacement of $\frac{s}{2}$ in position is the action of the operator $X(s) = e^{-is\hat{p}}$.

$$S(r) = e^{ir(\hat{x}\hat{p}+\hat{p}\hat{x})}, \tag{19}$$

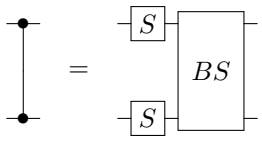
squeezes the position quadrature by r, while stretching the momentum quadrature by $1/r$.

$$P(s) = e^{is\hat{x}^2/2}, \tag{20}$$

shears a state along the position quadrature by a factor of s. When $s = 2t_2$ in the shearing operation we retrieve the third gate in the universal set.

$$B(\theta) = e^{i\theta(\hat{x}_1\hat{p}_2-\hat{p}_1\hat{x}_2)}, \tag{21}$$

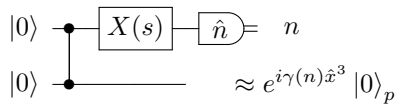
is a beamsplitter which acts on two modes and allows for multi-mode Gaussian operations. The final Gaussian element of the universal set in Eq. (7) is the Cz gate given by $e^{i\tau\hat{x}_1\hat{x}_2}$. This operation can be expressed as a combination of squeezing and beamsplitting in the following configuration:



(22)

where squeezing is denoted by S gates, and beam splitters by BS gates.

In order to implement higher-order gates we require an addition to the set of optical elements. The cubic phase operators are denoted by $V(t)$ gates in circuits, and are an example of these higher-order operations. To implement the cubic phase gate a photon counting measurement is needed, which introduces the higher-order non-linearity. The full implementation involves a displaced two-mode squeezed state for which $\hat{R}^\dagger \hat{n} \hat{R}$ (photon counting in a rotated basis) is measured on one arm. The desired cubic operation is then collapsed onto the second unmeasured mode [29]. This procedure is demonstrated in the following circuit:



(23)

where the initial states in both modes are squeezed momentum states. The states are entangled with a Cz gate and then a displacement operation is performed on the upper mode before a photon number measurement is made to collapse the cubic operation onto the bottom mode. This cubic state is approximate and depends on the measurement result n. The full implementation has also been demonstrated using repeat-until-success photon subtractions and Gaussian operations [30], as well as by using quadrature detection for feed-forward manipulation of parameters to produce nonlinear interaction [31].

While a photonic implementation of a CV quantum computer can operate at room temperature, the addition of sensitive detectors such as a photon counting measurement may require the use of refrigeration systems to improve accuracy. For example, an inaccurate detector might signal that it has detected a

photon when there were none, or miss the detection of a photon altogether. Other challenges and sources of error on a CV quantum computer include signal loss over larger distances, where the creation and use of a quantum repeater to is an ongoing challenge [22, 32], and the propagation of error due to the use of finite squeezing. For example the Cz gate implementation as in Eq. (22) assumes infinite squeezing and will retain additional error depending on the squeezing factor r in a realistic case where squeezing is finite. The addition of noise or error to a quantum system needs to be addressed in a practical quantum computer in order for it to be fault tolerant. This is done with sophisticated error correction schemes which have been studied extensively on CV quantum computers [20, 21, 33].

2 Methods for Exact Decompositions

This chapter follows closely and expands on the publication [4], for which I was the primary author. Here I describe a method to decompose multi-mode gates $e^{it\hat{H}}$, where the operator $\hat{H}$ is of the form

$$\hat{H} = \left(\prod_{j=1}^{N-1} \hat{x}_j \right) \hat{x}_N^n, \tag{24}$$

or

$$\hat{H} = \hat{x}_1^{n_1} \hat{x}_2^{n_2}, \tag{25}$$

for n, n_1, and n_2 positive integers. As well as single-mode gates $e^{it\hat{H}}$ with

$$\hat{H} = \hat{x}^N. \tag{26}$$

The label of the modes in Eq. (24) is arbitrary: the method works for any product where at most one operator has an exponent $n > 1$. In each case we require that N is divisible by either 2 or 3, and in the multi-mode case, the product nN must also be divisible by 2 or 3, as well as n_1 and n_2 divisible by 2. These gates can be extended to include momentum quadrature operators $\hat{p}_j$ by Fourier transforms acting on individual modes. As demonstrated in later sections and chapters, this set of gates for which exact

decompositions can be obtained encompasses a large class of operators arising in several CV quantum algorithms and simulations of bosonic systems.

The method relies on strategically employing: (i) unitary conjugation

$$U e^{it\hat{H}} U^\dagger = e^{itU\hat{H}U^\dagger}, \tag{27}$$

(ii) a lemma to the Baker-Campbell-Hausdorff (BCH) formula

$$e^{\hat{A}} \hat{B} e^{-\hat{A}} = \hat{B} + [\hat{A}, \hat{B}] + \frac{1}{2!}[\hat{A}, [\hat{A}, \hat{B}]] + \cdots, \tag{28}$$

and (iii), the identity

$$e^{i3\alpha^2 t \hat{p}_k \hat{x}_j^2} = e^{i2\alpha \hat{x}_j \hat{x}_k} e^{it\hat{p}_k^3} e^{-i\alpha \hat{x}_j \hat{x}_k} e^{-it\hat{p}_k^3} e^{-i2\alpha \hat{x}_j \hat{x}_k} e^{it\hat{p}_k^3} e^{i\alpha \hat{x}_j \hat{x}_k} e^{-it\hat{p}_k^3} e^{i\alpha^3 t \frac{3}{4} \hat{x}_j^3}, \tag{29}$$

with α and t real parameters. The proof of Eq. (29) uses both (i) and (ii) and follows closely the proof of Eq. (37) in the appendix. Before outlining the method in detail, some simple examples can be studied to illustrate the main idea behind the approach.

Suppose that the goal is to derive an exact decomposition for the unitary $e^{i\alpha \hat{x}_j \hat{x}_k \hat{x}_l}$. The first step of

the method is to express the operator $\hat{x}_j\hat{x}_k\hat{x}_l$ as a linear combination of polynomials of degree three in the quadrature operators $\hat{x}_j$, $\hat{x}_k$, and $\hat{x}_l$. Namely, one can employ the identity

$$\hat{x}_j\hat{x}_k\hat{x}_l = \tfrac{1}{6}[(\hat{x}_j+\hat{x}_k+\hat{x}_l)^3-(\hat{x}_j+\hat{x}_k)^3-(\hat{x}_j+\hat{x}_l)^3-(\hat{x}_k+\hat{x}_l)^3+\hat{x}_j^3+\hat{x}_k^3+\hat{x}_l^3], \tag{30}$$

which implies the identity

$$e^{i\alpha\hat{x}_j\hat{x}_k\hat{x}_l} = e^{\frac{i\alpha}{6}(\hat{x}_j+\hat{x}_k+\hat{x}_l)^3}e^{\frac{-i\alpha}{6}(\hat{x}_j+\hat{x}_k)^3}e^{\frac{-i\alpha}{6}(\hat{x}_j+\hat{x}_l)^3}e^{\frac{-i\alpha}{6}(\hat{x}_k+\hat{x}_l)^3}e^{\frac{i\alpha}{6}\hat{x}_j^3}e^{\frac{i\alpha}{6}\hat{x}_k^3}e^{\frac{i\alpha}{6}\hat{x}_l^3}, \tag{31}$$

since all the terms in the exponent commute. The right-hand side of this equation includes gates of the form $e^{\frac{i\alpha}{6}\hat{x}^3}$ that are part of the universal set, but it is still necessary to decompose the remaining terms. To do this, employ the decompositions

$$e^{i\alpha(\hat{x}_j+\hat{x}_k)^3} = e^{2i\hat{p}_j\hat{x}_k}e^{i\alpha\hat{x}_j^3}e^{-2i\hat{p}_j\hat{x}_k} \tag{32}$$

$$e^{i\alpha(\hat{x}_j+\hat{x}_k+\hat{x}_l)^3} = e^{2i\hat{p}_j\hat{x}_l}e^{i\alpha(\hat{x}_j+\hat{x}_k)^3}e^{-2i\hat{p}_j\hat{x}_l}, \tag{33}$$

which can be derived from Eqs. (27) and (28) using $U = e^{2i\hat{p}_j\hat{x}_k}$ as the unitary of conjugation. In summary, an exact decomposition can be derived by expressing $\hat{x}_j\hat{x}_k\hat{x}_l$ as a linear combination of polynomials of operators, allowing one to write the target gate $e^{i\alpha\hat{x}_j\hat{x}_k\hat{x}_l}$ in terms of a product of gates, each of which can be exactly decomposed.

Now suppose that the goal is to derive an exact decomposition for the higher-order single-mode gate $e^{i\alpha\hat{x}_j^4}$. Following the previous strategy, the operator $\hat{x}_j^4$ can be expressed as a linear combination of degree-four polynomials. More specifically, the identity

$$\hat{x}_j^4 = (\hat{x}_j^2 + \hat{x}_k)^2 - \hat{x}_k^2 - 2\hat{x}_j^2\hat{x}_k, \tag{34}$$

implies the relation

$$e^{i\alpha\hat{x}_j^4} = e^{i\alpha(\hat{x}_j^2+\hat{x}_k)^2} e^{-i\alpha\hat{x}_k^2} e^{-2i\alpha\hat{x}_j^2\hat{x}_k}. \tag{35}$$

Here, the gate $e^{-i\alpha\hat{x}_k^2}$ is part of the universal set, while Eq. (29) gives an exact decomposition for $e^{-i\alpha\hat{x}_j^2\hat{x}_k}$ up to a Fourier transform. As before, the remaining term can be decomposed using unitary conjugation:

$$e^{i\alpha(\hat{x}_j^2+\hat{x}_k)^2} = e^{2i\hat{p}_k\hat{x}_j^2} e^{i\alpha\hat{x}_k^2} e^{-2i\hat{p}_k\hat{x}_j^2}, \tag{36}$$

leading to a full decomposition for the target gate $e^{i\alpha\hat{x}_j^4}$. Note that an additional ancillary mode k was required in this decomposition.

To extend this method to a more general setting, the same basic strategy can be employed: express the target gate in terms of a linear combination of polynomials and decompose the resulting gates in terms of unitary conjugation or previously derived decompositions.

2.1 Single-Mode Gates

The following describes the method for decomposing single-mode gates of the form $e^{i\alpha\hat{x}^N}$ with N an integer divisible by 2 or 3. In the previous example, we showed how Eq. (29) could be employed to decompose $e^{i\alpha\hat{x}^4}$. Generalizing Eq. (29) to higher order similarly enables decompositions of single-mode gates with larger exponents. It can be shown that such a general form exists, given by the expression

$$e^{2i\alpha^2\hat{p}_k\hat{x}_j^N} = e^{2i\alpha\hat{x}_j^{N-2}\hat{x}_k}e^{-i\alpha\hat{x}_j^2\hat{p}_k^2}e^{-2i\alpha\hat{x}_j^{N-2}\hat{x}_k}e^{i\alpha\hat{x}_j^2\hat{p}_k^2}e^{i\alpha^3\hat{x}_j^{2(N-1)}}, \tag{37}$$

for $N \geq 2$. The proof of this formula can be found in the Appendix. This formula holds with the addition of another mode and can be proven in a similar manner.

$$e^{2i\alpha^2\hat{p}_k\hat{p}_l\hat{x}_j^n} = e^{2i\alpha\hat{x}_j^{n-2}\hat{x}_k\hat{x}_l}e^{-i\alpha\hat{x}_j^2\hat{p}_k^2}e^{-2i\alpha\hat{x}_j^{n-2}\hat{x}_k\hat{x}_l}e^{i\alpha\hat{x}_j^2\hat{p}_k^2}e^{i\alpha^3\hat{x}_j^{2(n-1)}\hat{p}_l}. \tag{38}$$

These decompositions require the gate $e^{i\alpha\hat{x}_j^2\hat{x}_k^2}$, which is not part of the universal set. However, an exact decomposition also holds for this gate (see the Appendix for a proof):

$$e^{i\alpha\hat{x}_j^2\hat{x}_k^2} = e^{i2\hat{p}_j\hat{x}_k}e^{i\frac{\alpha}{12}\hat{x}_j^4}e^{-i4\hat{p}_j\hat{x}_k}e^{i\frac{\alpha}{12}\hat{x}_j^4}e^{i2\hat{p}_j\hat{x}_k}e^{-i\frac{\alpha}{6}\hat{x}_j^4}e^{-i\frac{\alpha}{6}\hat{x}_k^4}, \tag{39}$$

where we can employ the previously derived decomposition for $e^{i\hat{x}_j^4}$. The form of Eq. (39) can be expanded to create more arbitrary two-mode gates with higher order as in Eq. (25), the details of which will be shown in the following section on multi-mode decompositions. To obtain a general form for single-mode decompositions, Eq. (39) is used as well as the fourth-order single-mode decomposition in Eq. (35) to first obtain a higher-order version of Eq. (29):

$$e^{2i\alpha^2\hat{p}_k\hat{x}_j^3} = e^{2i\alpha\hat{x}_j\hat{x}_k}e^{-i\alpha\hat{x}_j^2\hat{p}_k^2}e^{-2i\alpha\hat{x}_j\hat{x}_k}e^{i\alpha\hat{x}_j^2\hat{p}_k^2}e^{-2i\alpha^3\hat{x}_j^4}. \tag{40}$$

This can then be used to create a decomposition for $e^{i\hat{x}_j^6}$ in a similar way to the decomposition of the gate $e^{i\hat{x}_j^4}$. The decomposition for $e^{i\hat{x}_j^6}$ can once more be combined with Eq. (39) to derive an exact decomposition for the next highest power of the two-mode gate, namely $e^{2i\alpha^2\hat{p}_k\hat{x}_j^4}$. This process can be continued until the general recursive form in Eq. (37) is reached, as well as a more general decomposition of single-mode operations:

$$e^{i\alpha\hat{x}_k^N} = e^{2i\hat{p}_j\hat{x}_k^{N/2}}e^{i\alpha\hat{x}_j^2}e^{-2i\hat{p}_j\hat{x}_k^{N/2}}e^{-i\alpha\hat{x}_j^2}e^{-2i\alpha\hat{x}_j\hat{x}_k^{N/2}}, \tag{41}$$

that holds when N is even. The proof of this equation is detailed in the Appendix, but follows similar steps to the fourth-order single-mode gate in Eq. (35). If N is odd and a multiple of three, exact decompositions

can also be derived by noting the following relation:

$$2\hat{x}_k^N = 2\left(\hat{x}_j + \hat{x}_k^{N/3}\right)^3 - 3\left(\hat{x}_l + \hat{x}_j^2 + \hat{x}_k^{N/3}\right)^2 - 2\hat{x}_j^3 + 3\hat{x}_j^4 + 3\hat{x}_k^{2N/3} - 6\hat{x}_j\hat{x}_k^{2N/3} + 6\hat{x}_j^2\hat{x}_l + 6\hat{x}_k^{N/3}\hat{x}_l + 3\hat{x}_l^2. \tag{42}$$

Therefore, for N odd and divisible by 3, we can decompose the single-mode operation as

$$e^{i2\alpha\hat{x}_k^N} = e^{i2\alpha\left(\hat{x}_j+\hat{x}_k^{N/3}\right)^3} e^{-i3\alpha\left(\hat{x}_l+\hat{x}_j^2+\hat{x}_k^{N/3}\right)^2} e^{-i2\alpha\hat{x}_j^3} e^{i3\alpha\hat{x}_j^4} e^{i3\alpha\hat{x}_k^{2N/3}} e^{-i6\alpha\hat{x}_j\hat{x}_k^{2N/3}} e^{i6\alpha\hat{x}_j^2\hat{x}_l} e^{i6\alpha\hat{x}_k^{N/3}\hat{x}_l} e^{i3\alpha\hat{x}_l^2}. \tag{43}$$

Here, the gates $e^{i2\alpha\left(\hat{x}_j+\hat{x}_k^{N/3}\right)^3}$ and $e^{-i3\alpha\left(\hat{x}_l+\hat{x}_j^2+\hat{x}_k^{N/3}\right)^2}$ can be decomposed using the expressions

$$e^{i2\alpha\left(\hat{x}_j+\hat{x}_k^{N/3}\right)^3} = e^{2i\hat{p}_j\hat{x}_k^{N/3}} e^{i2\alpha\hat{x}_j^3} e^{-2i\hat{p}_j\hat{x}_k^{N/3}}, \tag{44}$$

$$e^{-i3\alpha\left(\hat{x}_l+\hat{x}_j^2+\hat{x}_k^{N/3}\right)^2} = e^{2i\hat{p}_l\hat{x}_k^{N/3}} e^{2i\hat{p}_l\hat{x}_j^2} e^{-i3\alpha\hat{x}_l^2} e^{-2i\hat{p}_l\hat{x}_j^2} e^{-2i\hat{p}_l\hat{x}_k^{N/3}}, \tag{45}$$

which as before are obtained using unitary conjugation. The other gates in Eq. (43) can be decomposed with the previous general formulas Eq. (41) and Eq. (37). It is important to note that for even N only one ancillary mode is needed, whereas for N odd and a multiple of three, two additional ancillary modes are needed. In the case of multi-mode gates the number of ancillary modes depends only on the number of single-mode gates in the decomposition which further need to be decomposed with Eq.(41) or Eq.(43).

In either case only one or two ancillary modes are needed because these modes may be repurposed for each single-mode gate that appears in the multi-mode decomposition.

2.2 Multi-Mode Gates

The multi-mode case is studied where $\hat{H}$ is given by

$$\hat{H} = \prod_{j=1}^{N} \hat{x}_j^{n_j}, \tag{46}$$

where the n_j are positive integers. It is discussed later why restrictions are necessary on the exponents n_j, leading to exact decompositions for operators as in Eq. (24), as well as how to form decompostions as in Eq. (25).

2.2.1 Multi-Mode Gates With More Than Two Modes

As discussed previously, the first step to decompose a multi-mode gate $e^{it\hat{H}}$ is to express $\hat{H}$ as a linear combination of operators. Let $[N]^k$ be the set of all k-subsets of $\{1, 2, \ldots, N\}$, i.e., all subsets containing k elements. For example, $[3]^2 = \{\{1,2\},\{1,3\},\{2,3\}\}$. The goal is to find coefficients $c_1, c_2, \ldots, c_N$ such

that [34]

$$\prod_{j=1}^{N} \hat{x}_j^{n_j} = \sum_{k=1}^{N} c_k \sum_{S \in [N]^k} \left(\sum_{i=1}^{k} \hat{x}_{S_i}^{n_{S_i}} \right)^N , \tag{47}$$

where $S \in [N]^k = \{S_1, S_2, \ldots, S_k\}$. When expanded, the term on the right-hand side contains several monomials of the position operators, including the desired term $\prod_{j=1}^{N} \hat{x}_j^{n_j}$. Each monomial is multiplied by a factor that is a linear combination of the coefficients c_k, and the goal is to set these factors to zero for all monomials except $\prod_{j=1}^{N} \hat{x}_j^{n_j}$. As shown in the Appendix, this gives rise to a linear system of equations for the coefficients c_k such that Eq. (47) holds whenever the coefficients $\vec{c} = (c_N, c_{N-1}, \ldots, c_1)$ satisfy the linear system $A\vec{c} = 0$. The matrix A is a rectangular matrix which is independent of the exponents n_j and is given by

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & \ldots & 0 \\ 1 & 2 & 1 & 0 & \ldots & 0 \\ 1 & 3 & 3 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \binom{N-1}{0} & \binom{N-1}{1} & \binom{N-1}{2} & \binom{N-1}{3} & \cdots & \binom{N-1}{N-1} \end{pmatrix}, \tag{48}$$

i.e., the coefficients of A follow the structure of Pascal's triangle. The formula for each element of matrix A is

$$\begin{aligned} A_{i,j} &= \ 0, && \text{if} \quad i < j-1 \\ &= \frac{i!}{(j-1)!(i-j+1)!}, && \text{otherwise.} \end{aligned} \tag{49}$$

Note that because this linear system is underdetermined since there are $N-1$ equations for N variables. However, by fixing c_N, it is possible to find a specific non-trivial solution, as shown in the following observation.

Observation 1. *A solution to the linear system $A\vec{c} = 0$ with $\vec{c} = (c_N, c_{N-1}, \ldots, c_1)$ and A as in Eq. (48) is given by $c_{N-k} = (-1)^k c_N$.*

Proof. For simplicity and without loss of generality, let $c_N = 1$. The base case for $N = 2$ is trivially true; it is simply $c_2 + c_1 = 0 \implies c_1 = -1$. Now examine the general structure for the case with $N = k$. Assume that the claimed solution $c_{N-k} = (-1)^k$ with $k = 0, 1, \ldots, N-2$ is true for $N = K-1$, i.e., the system when the last row and last column are omitted from the matrix A. For the case $N = K$, the last

row of A determines an equation for the remaining coefficient c_1. We then have

$$\sum_{k=0}^{K-1} \binom{K-1}{k} c_k = 0 \tag{50}$$

$$= \sum_{\ell=0}^{K-1} \binom{K-1}{K-\ell} c_{K-\ell}$$

$$= c_1 + \sum_{\ell=0}^{K-2} \binom{K-1}{K-\ell} (-1)^\ell = 0. \tag{51}$$

We want to show that $c_1 = (-1)^{K-1}$ is a solution to this equation. This yields

$$(-1)^{K-1} + \sum_{\ell=0}^{K-2} \binom{K-1}{K-\ell} (-1)^\ell = \sum_{\ell=0}^{K-1} \binom{K-1}{K-\ell} (-1)^\ell$$

$$= (-1+1)^{K-1} = 0 \tag{52}$$

as desired, where the last line follows from the binomial theorem. □

The solution $c_{N-k} = (-1)^k c_N$ is valid for any value of c_N. In order to satisfy Eq. (47) exactly, we simply fix $c_N = 1/N!$. With this choice of coefficients c_k, the sum of polynomials on the right-hand side of Eq. (47) is exactly equal to the multi-mode product of operators on the left-hand side. Thus, the process for decomposing multi-mode gates is to find an exact decomposition for each polynomial appearing on the right-hand side of Eq. (47). As done before, specifically in Eqs. (32), (33), (36), (44), and (45), decomposition of polynomials is performed using unitary conjugation with the gate $e^{2i\hat{p}_1 \hat{x}_j^{n_j}}$ –

with decomposition in Eq. (37) – and the lemma to the BCH formula. More precisely, we employ the following identity to decompose an arbitrary polynomial:

$$e^{it(\hat{x}_1+\hat{x}_2^{n_2}+\hat{x}_3^{n_3}+\cdots+\hat{x}_m^{n_m})^N} = e^{2i\hat{p}_1\hat{x}_m^{n_m}}\cdots e^{2i\hat{p}_1\hat{x}_3^{n_3}}e^{2i\hat{p}_1\hat{x}_2^{n_2}}e^{it\hat{x}_1^N}e^{-2i\hat{p}_1\hat{x}_2^{n_2}}e^{-2i\hat{p}_1\hat{x}_3^{n_3}}\cdots e^{-2i\hat{p}_1\hat{x}_m^{n_m}}. \tag{53}$$

Using Eq. (53), it is not possible to find exact decompositions for all operators $\prod_{j=1}^{N}\hat{x}_j^{n_j}$ in Eq. (46), as there are restrictions on the exponents n_j. The restrictions are as follows:

1. There can exist at most one j such that $n_j \neq 1$. This restriction arises from the fact that the central operator in Eq. (53), namely $\hat{x}_1$, must have an exponent equal to one. Therefore, in order to use Eq. (53) to decompose every polynomial $\left(\sum_{i=1}^{k}\hat{x}_{S_i}^{n_{S_i}}\right)^N$ on the right-hand side of Eq. (47), all k-subsets S with $k > 1$ must contain at least one element $S_i \in S$ such that $n_{S_i} = 1$. This is only possible if there exists at most one j such that $n_j \neq 1$.

2. The product Nn_j must be divisible by either 2 or 3 for all j. This arises because the $k = 1$ terms in Eq. (47) produce monomials that include only single-mode operators to the power of Nn_j. As shown in the previous section, the method only produces exact decompositions for single-mode operations with power divisible by 2 or 3.

To summarize, we employ Eq. (47) to express a multi-mode operator as a linear combination of polynomials. Each polynomial can then be exactly decomposed using Eq. (53) and single-mode decompositions from the previous section. This yields a method for constructing exact decompositions of operators of the

form $e^{it\hat{H}}$, for $\hat{H} = \left(\prod_{j=1}^{N-1} \hat{x}_j\right) \hat{x}_N^n$, with both Nn and N divisible by either 2 or 3. In order to construct multi-mode gates where more then one mode has its power raised to an integer greater then one, as in Eq. (25), an iterative process can be used that follows from the construction of Eq. (39).

2.2.2 Two-Mode Gates With Higher Powers

As shown in the appendix, Eq. (39) is constructed by using unitary conjugation to create the polynomials $(\hat{x}_j + \hat{x}_k)^4 + (\hat{x}_j - \hat{x}_k)^4$ which when added together cancel all of the terms with odd powers in $\hat{x}_k$. This structure can be exploited even when Eq. (37) is used in unitary conjugation to raise the power of $\hat{x}_k$ in the brackets. For example, note the following equation

$$\left(\hat{x}_j + \hat{x}_k^{b/2}\right)^4 + \left(\hat{x}_j - \hat{x}_k^{b/2}\right)^4 = 2\hat{x}_j^4 + 12\hat{x}_j^2\hat{x}_k^b + 2\hat{x}_k^{2b} \tag{54}$$

where b is an even number. This implies an exact decomposition for $e^{it\hat{x}_j^2\hat{x}_k^b}$ can be derived, as both $e^{it\left(\hat{x}_j+\hat{x}_k^{b/2}\right)^4}$ and $e^{it\left(\hat{x}_j-\hat{x}_k^{b/2}\right)^4}$ can be decomposed using Eq. (37) in unitary conjugation:

$$e^{it\left(\hat{x}_j+\hat{x}_k^{b/2}\right)^4} = e^{2i\hat{p}_j\hat{x}_k^{b/2}} e^{it\hat{x}_j^4} e^{-2i\hat{p}_j\hat{x}_k^{b/2}}, \tag{55}$$

$$e^{it\left(\hat{x}_j-\hat{x}_k^{b/2}\right)^4} = e^{-2i\hat{p}_j\hat{x}_k^{b/2}} e^{it\hat{x}_j^4} e^{2i\hat{p}_j\hat{x}_k^{b/2}}. \tag{56}$$

Therefore, the exact decomposition for $e^{it\hat{x}_j^2\hat{x}_k^b}$ is given by

$$e^{it\hat{x}_j^2\hat{x}_k^b} = e^{2i\hat{p}_j\hat{x}_k^{b/2}} e^{i\frac{t}{12}\hat{x}_j^4} e^{-2i\hat{p}_j\hat{x}_k^{b/2}} e^{-2i\hat{p}_j\hat{x}_k^{b/2}} e^{i\frac{t}{12}\hat{x}_j^4} e^{2i\hat{p}_j\hat{x}_k^{b/2}} e^{-i\frac{t}{6}\hat{x}_j^4} e^{-i\frac{t}{6}\hat{x}_k^{2b}}, \tag{57}$$

where the decompositions for the single mode gates are given in Eq. (41). This result can be further generalized to gates of the form $e^{it\hat{x}_j^a\hat{x}_k^b}$, with a and b even, by increasing the power of the jth mode. Note the following polynomial equation

$$\left(\hat{x}_j + \hat{x}_k^{b/2}\right)^{a+2} + \left(\hat{x}_j - \hat{x}_k^{b/2}\right)^{a+2} = 2\hat{x}_j^{a+2} + 2\binom{a+2}{a}\hat{x}_j^a\hat{x}_k^b + 2\binom{a+2}{a-2}\hat{x}_j^{a-2}\hat{x}_k^{2b} + 2\binom{a+2}{a-4}\hat{x}_j^{a-4}\hat{x}_k^{3b} + \cdots + 2\hat{x}_k^{(a+2)b/2}, \tag{58}$$

which implies the operator relation

$$\begin{aligned} e^{it2\binom{a+2}{a}\hat{x}_j^a\hat{x}_k^b} = {} & e^{it\left(\hat{x}_j+\hat{x}_k^{b/2}\right)^{a+2}} e^{it\left(\hat{x}_j-\hat{x}_k^{b/2}\right)^{a+2}} e^{-it2\hat{x}_j^{a+2}} e^{-it2\binom{a+2}{a-2}\hat{x}_j^{a-2}\hat{x}_k^{2b}} e^{-it2\binom{a+2}{a-4}\hat{x}_j^{a-4}\hat{x}_k^{3b}} \\ & \cdots e^{-it2\binom{a+2}{2}\hat{x}_j^2\hat{x}_k^{ab/2}} e^{-it2\hat{x}_k^{ab/2+b}}. \end{aligned} \tag{59}$$

The gates $e^{it\left(\hat{x}_j+\hat{x}_k^{b/2}\right)^{a+2}}$ and $e^{it\left(\hat{x}_j-\hat{x}_k^{b/2}\right)^{a+2}}$ are decomposed similarly to Eq. (55) but with the jth mode raised to the power $a+2$ instead of 4 on the right hand side. The single mode gates are all raised to even powers and as such can be decomposed exactly with Eq. (41). The decompositions for the multi-mode gates on the right hand side follow recursively starting with the rightmost gate. $e^{-it2\binom{a+2}{2}\hat{x}_j^2\hat{x}_k^{ab/2}}$ can

be decomposed with the less general form in Eq. (57) as both a and b are even. The next gate in the sequence, namely, $e^{-it2\binom{a+2}{4}\hat{x}_j^4\hat{x}_k^{ab/2-b}}$ can be decomposed with the above equation by substituting $a' = 4$ and $b' = ab/2 - b$. Each multi-mode gate up to $e^{-it2\binom{a+2}{a-2}\hat{x}_j^{a-2}\hat{x}_k^{2b}}$ can be substituted back into Eq. (59) in this way and will only need decompositions of the multi-mode gates to the right of itself, as well as gates covered by Eq. (57).

For example, to decompose $e^{it\hat{x}_j^6\hat{x}_k^6}$ from Eq. (59) we would need multi-mode decompositions for $e^{it\hat{x}_j^4\hat{x}_k^{12}}$ and $e^{it\hat{x}_j^2\hat{x}_k^{18}}$, where substituting $e^{it\hat{x}_j^4\hat{x}_k^{12}}$ back into Eq. (59) also needs a multi-mode decomposition for $e^{it\hat{x}_j^2\hat{x}_k^{24}}$. Both $e^{it\hat{x}_j^2\hat{x}_k^{18}}$ and $e^{it\hat{x}_j^2\hat{x}_k^{24}}$ can then be decomposed by using Eq. (57). In every case the remaining single mode gates will have even power and as such can be decomposed.

Note that the recursion in decompositions of this form can form much higher order single mode gates. As a result the gate counts for this part of the method may scale poorly with respect to the power of each mode. For example the gate count for the exact decomposition of the sixth order gate $e^{it\hat{x}_j^2\hat{x}_k^4}$ is 3320 gates from the universal set, whereas the single mode sixth order gate $e^{it\hat{x}^6}$ requires almost four times fewer.

Table 1 shows a comparison of gate counts for a variety of gates which can be decomposed using the methods described in this chapter. It also includes a comparison of exact decompositions with the standard commutator approximation, where the gate counts for the commutator approximations are taken to a precision of 10^{-3}. The two gates in the table with the lowest gate counts in the exact method are the third-order three-mode gate and the fourth-order single-mode gate, with 17 and 29 gates in their

respective decompositions. This is in contrast to the commutator approximation where the addition of the third mode greatly increases the circuit depth. The structure of each method seems to indicate that the exact decompositions scale better under addition of more modes. Also, the need to repeat the set of gates to improve precision in the commutator approximation produces several orders-of-magnitudes increase in the resulting circuit depths.

Table 2 demonstrates the applicability of the operations shown in Table 1 by providing example algorithms for which they appear in the decomposition. In some cases, only part of a desired operation might be decomposed exactly, but as demonstrated in Table 1, the exact decompositions even for these portions can produce a significant decrease in circuit depth. The final entry in the table contains a general operation that depends on the choice of $h(\hat{x}_1)$, which is chosen to be a polynomial in $\hat{x}_1$. This operation will be covered by the exact decomposition method regardless of the choice of $h(\hat{x}_1)$ because there are four total modes. Assuming $h(\hat{x}_1) = \hat{x}_1^n$, then $Nn = 4n$ is always even and therefore the single mode operation $e^{it\hat{x}_1^{4n}}$ can be decomposed exactly. Also, since the final three modes are all to unit power, any one of them may be used as the exponent of the central operator in unitary conjugation. Therefore both of the restrictions of the method have been met regardless of n. By linearity, the same holds for a general polynomial $h(\hat{x}_1) = \sum_n a_n \hat{x}_1^n$.

Target gate	Commutator approx. (10^{-3} precision)	Exact decomposition	Cubic phase gates in each exact decomp.
$e^{it\hat{x}^4}$	1.8×10^4 gates	29 gates	15 gates
$e^{it\hat{x}_j^2\hat{x}_k^2}$	2.8×10^4 gates	119 gates	60 gates
$e^{it\hat{x}_j\hat{x}_k^3}$	2.9×10^8 gates	269 gates	135 gates
$e^{it\hat{x}_j\hat{x}_k\hat{x}_l}$	4.2×10^8 gates	17 gates	7 gates
$e^{it\hat{x}_j^2\hat{x}_k\hat{x}_l}$	1.4×10^9 gates	249 gates	125 gates
$e^{it\hat{x}_j\hat{x}_k\hat{x}_l\hat{x}_m}$	6.9×10^{13} gates	440 gates	225 gates
$e^{it\hat{x}^6}$	1.2×10^{13} gates	809 gates	405 gates
$e^{it\hat{x}_j^2\hat{x}_k^4}$	2.4×10^{13} gates	3320 gates	1670 gates

Table 1: Gate counts for decompositions of some common operations, using the standard commutator approximation as well as the exact decompositions described in this chapter. The gate counts neglect any Fourier transforms used by either method as they are inexpensive to implement experimentally. The final column includes counts of cubic phase gates needed in each exact decomposition.

Algorithm	Hamiltonian and Operators covered by method	Circuit depth of operator
Vibrational dynamics of molecules, Ref. [35]	$\hat{H} = \hbar \sum_{i \leq j} \frac{x_{ij}}{2} \sqrt{\omega_i \omega_j} \left(\hat{a}_i^\dagger \hat{a}_i + \hat{a}_j^\dagger \hat{a}_j + 2 \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_i \hat{a}_j \right)$ $e^{it\hat{H}}$ contains operator $e^{it\hat{x}_j^2 \hat{x}_k^2}$	119 gates
Non-homogeneous linear PDEs, Ref. [7]	$\hat{H} = \sum_{j=1}^{N} \left(a_j \hat{x}_j + b_j \hat{p}_j + \alpha_j \hat{x}_j^2 + \beta_j \hat{p}_j^2 \right) \hat{x}_k \hat{x}_l$ $e^{it\hat{H}}$ contains $e^{it\hat{x}_j \hat{x}_k \hat{x}_l}$ and $e^{it\hat{x}_j^2 \hat{x}_k \hat{x}_l}$	17 gates and 249 gates
Dipole interaction term of Bose Hubbard, Ref. [6]	$\hat{H} = V \sum_{i \leq j} \hat{a}_i^\dagger \hat{a}_i \hat{a}_j^\dagger \hat{a}_j$ $e^{it\hat{H}}$ contains the operator $e^{it\hat{x}_i^2 \hat{x}_j^2}$	119 gates
One particle tunneling of Bose Hubbard, Ref. [36]	$\hat{H} = -T \sum_{i \leq j} \hat{a}_i^\dagger \left(\hat{n}_i + \hat{n}_j \right) \hat{a}_j$ $e^{it\hat{H}}$ contains the operator $e^{it\hat{x}_i \hat{x}_j^3}$	125 gates
Nearest-neighbor tunneling of Bose Hubbard, Ref. [36]	$\hat{H} = \frac{P}{2} \sum_{i \leq j} \hat{a}_i^\dagger \hat{a}_i^\dagger \hat{a}_j \hat{a}_j$ $e^{it\hat{H}}$ contains the operator $e^{it\hat{x}_i^2 \hat{x}_j^2}$	119 gates
Cross-Kerr Hamiltonian, Ref. [3]	$\hat{H} = \left(\hat{x}_i^2 + \hat{p}_i^2 \right) \otimes \left(\hat{x}_j^2 + \hat{p}_j^2 \right)$ $e^{it\hat{H}}$ contains the operator $e^{it\hat{x}_i^2 \hat{x}_j^2}$	119 gates
Principal component analysis, Ref. [8]	$R(\hat{p}_R) = e^{i\delta \hat{p}_R (\hat{a}_1 \hat{a}_2^\dagger + \hat{a}_1^\dagger \hat{a}_2)}$ $R(\hat{p}_R)$ contains the operator $e^{i\delta \hat{x}_R \hat{x}_1 \hat{x}_2}$	17 gates
Matrix inversion algorithm, Ref. [8]	$R(\hat{p}_R \hat{p}_S) = e^{i\gamma \hat{p}_R \hat{p}_S (\hat{a}_1 \hat{a}_2^\dagger + \hat{a}_1^\dagger \hat{a}_2)}$ $R(\hat{p}_R \hat{p}_S)$ contains the operator $e^{i\gamma \hat{x}_R \hat{x}_S \hat{x}_1 \hat{x}_2}$	440 gates
Monte Carlo integration, Ref. [37]	$e^{ih(\hat{x}_1) \hat{p}_2 \hat{p}_3 \hat{p}_\phi}$, decomposed exactly for any $h(\hat{x}_1)$ polynomial in $\hat{x}_1$	(see Table 3)

Table 2: Algorithms which require a decomposition for gates covered by the exact method. The second column shows the operator or Hamiltonian that appears in the algorithm as well as the portion which is covered by the exact method. The final column gives the gate count of the portion which can be decomposed exactly.

3 The Bose-Hubbard Model

In order to demonstrate the use of known decomposition techniques as well as some exact decompositions, we will look at an example of a quantum simulation of a physical system. Quantum simulation of physical systems constitutes an important application for early quantum computing devices [2, 38, 39]. A quantum computer can be used for the purpose of observing properties of that system which may be hard to obtain from direct experiments or classical computing. For example, such simulations may be used to determine the ground state energies of certain molecules or to simulate systems of molecules, which can be difficult to determine using a classical computer [40–42].

Usually, the starting point is a reasonable model for the Hamiltonian of the physical system and mapping of that Hamiltonian into the degrees of freedom of the quantum simulator. Once a suitable mapping from the physical system has been found, the Hamiltonian time evolution operator is simulated by applying specific operations on the quantum device. The domain of Hamiltonian simulation examines the efficient implementation of Hamiltonians by considering their properties such as locality or sparsity. Often such simulations are performed efficiently, that is polylogarithmically in the size of the Hilbert space and close to linear in the simulation time. For qubit quantum computers, such Hamiltonian simulations have been discussed in detail in [27, 43–50].

In this chapter I follow closely the publication [6] which examines the Bose-Hubbard system in the context of quantum simulation. The Bose-Hubbard model is one that has been studied extensively,

describing a system of bosonic particles trapped in an optical lattice [36]. This model is simulated using various methods such as quantum Monte Carlo simulations [51–55]. The purpose of most of these simulations has been to examine state transitions between a superfluid and a Mott insulator [36,52,54–56]. The Bose-Hubbard model also has applications in examining the generation of entanglement [57] and the creation of quantum magnetic insulators [58]. It has been shown that the one-dimensional Bose-Hubbard may be easily simulated classically [59,60]. However, the general problem of finding the ground state of a quantum system, including the Bose-Hubbard quantum system, is part of the QMA-complete (quantum Merlin Arthur) complexity class. This is the set of decision problems on a quantum computer for which there exists a polynomial-time quantum verifier, and that every QMA problem can be reduced to it. This is the quantum analogue to the NP (non-deterministic polynomial time) class of problems. As well, simulating the time evolution operator of the Bose-Hubbard system is BQP-complete (bounded-error polynomial time) when formalized as a decision problem [50, 61–63]. Analogous to the classical complexity class P (polynomial time), BQP problems, are decision problems which can be solved by a quantum computer in polynomial time. Similarly BQP-complete means that a problem is in BQP and every BQP problem can be reduced to it. For the Bose-Hubbard system this means that there exists an efficient quantum algorithm that can accurately determine whether or not a given output was one produced from the system itself, whereas it is believed that no such efficient classical algorithm exists. Here, 'efficient' means that the algorithm scales as a polynomial in the size of the system.

For the Bose-Hubbard Hamiltonian, I show that a CV system allows for a straightforward mathematical

decomposition into the required logic gates, as well as a circuit topology that allows for advantages in implementation. I present the exact resource counts required to simulate the Bose-Hubbard Hamiltonian on a CV quantum computer as well as the circuits that implement 1-D and 2-D Bose-Hubbard models of variable sizes.

3.1 Gate Decomposition of Bose-Hubbard Hamiltonian

The Bose-Hubbard Hamiltonian describes a system of bosonic particles trapped in an optical lattice of N sites. Using notation from [36], it is given by

$$H = -\frac{J}{2}\sum_{\{i,j\}} \hat{a}_i^\dagger \hat{a}_j + \frac{U}{2}\sum_{i=1}^{N} \hat{n}_i(\hat{n}_i - 1), \tag{60}$$

where the two terms with the factors J and U represent the tunneling of a particle from one site to a neighboring site, and the on-site interaction, respectively (see Fig. 1 for a schematic). The bosonic creation (annihilation) operators are given by $\hat{a}_i^\dagger$ ($\hat{a}_i$) and the number operator is $\hat{n}_i = \hat{a}_i^\dagger \hat{a}_i$. The sum $\sum_{\{i,j\}}$ spans neighboring sites. Additional terms may be added to the Hamiltonian which come from dipole interactions [36], but first the terms in Eq. (60) are examined in detail. The objective of the gate decomposition is to find an appropriate implementation of quantum gates which can be used to simulate the evolution of this Hamiltonian e^{itH} for a time t. In order to do this, e^{itH} is decomposed into more elementary time evolution operators. Note that the physical time evolution operator here would be

e^{-2itH} to be consistent with the choice of $\hbar$, but for simplicity I will use e^{itH} throughout this chapter. In the Hamiltonian, the J terms as well as part of the U terms are of Gaussian order, therefore they may be efficiently implemented with linear optics. The non-Gaussian U term may be further broken down using the exact decomposition techniques in the previous section. This decomposition is now examined more precisely. First, the operators $\hat{a}_i^\dagger$, $\hat{a}_i$ and $\hat{n}_i$ are expanded in terms of position operators $\hat{x}_i$ and momentum operators $\hat{p}_i$ via Eqs. (2), (3)

$$\begin{aligned} \hat{a}_i &= \hat{x}_i + i\hat{p}_i, \\ \hat{a}_i^\dagger &= \hat{x}_i - i\hat{p}_i, \\ \hat{a}_i^\dagger \hat{a}_i &= \hat{x}_i^2 + \hat{p}_i^2 + i[\hat{x}_i, \hat{p}_i], \end{aligned} \tag{61}$$

where as before $[\hat{x}_i, \hat{p}_i] = \frac{i}{2}$. Considering these relations and neglecting a constant energy shift an expanded Hamiltonian is then written as

$$H = -J \sum_{\{i,j\}:i<j} (\hat{x}_i\hat{x}_j + \hat{p}_i\hat{p}_j) + \frac{U}{2} \sum_i \left(\left(\hat{x}_i^4 + \hat{x}_i^2\hat{p}_i^2 + \hat{p}_i^2\hat{x}_i^2 + \hat{p}_i^4 - \hat{x}_i^2 - \hat{p}_i^2\right) + \left(-\hat{x}_i^2 - \hat{p}_i^2\right) \right). \tag{62}$$

We can simplify $\hat{x}_i^2\hat{p}_i^2 + \hat{p}_i^2\hat{x}_i^2$ with a relation from [3]

$$\hat{x}_i^2\hat{p}_i^2 + \hat{p}_i^2\hat{x}_i^2 = -\frac{4}{9} i[\hat{x}_i^3, \hat{p}_i^3]. \tag{63}$$

As the time evolution to be simulated is e^{itH}, we can use the Trotter-Suzuki formula as in Eq. (13) where we refer to the remainder term as $\mathcal{R}$. The choice of K controls the size of the remainder $\mathcal{R}$ and thus gives the accuracy of the decomposition. The size of the remainder can be bounded by [27]

$$\|\mathcal{R}\| = O\left(\frac{N^2 t^2 \Lambda^2}{K}\right), \tag{64}$$

where $\Lambda := \max_j \|H_j\|$ is the largest Hamiltonian norm. In our case, we can write

$$e^{itH} = \left(\prod_{\{i,j\}:i<j} e^{-i\frac{t}{K}J\hat{x}_i\hat{x}_j} e^{-i\frac{t}{K}J\hat{p}_i\hat{p}_j} \prod_i e^{i\frac{t}{K}\frac{U}{2}\hat{x}_i^4} e^{\frac{t}{K}\frac{2U}{9}[\hat{x}_i^3,\hat{p}_i^3]} e^{i\frac{t}{K}\frac{U}{2}\hat{p}_i^4} e^{-i\frac{t}{K}U\hat{x}_i^2} e^{-i\frac{t}{K}U\hat{p}_i^2} \right)^K + \mathcal{R}. \tag{65}$$

The largest Hamiltonian norm here is at most $\Lambda = O(\text{poly}(J,U))$, taken to be $O(1)$, as all terms involve the position and momentum operators [3].

We can rotate every momentum operator into the position basis by a Fourier transform. For every polynomial g we have

$$g(\hat{p}_i) = g(\mathcal{F}_i \hat{x}_i \mathcal{F}_i^\dagger) = \mathcal{F}_i g(\hat{x}_i) \mathcal{F}_i^\dagger. \tag{66}$$

In addition, we can use the standard commutator approximation via the relation [3]

$$e^{[A,B]\tau^2} = e^{iB\tau} e^{iA\tau} e^{-iB\tau} e^{-iA\tau} e^{iB\tau} e^{iA\tau} e^{-iB\tau} e^{-iA\tau} + O(\tau^4), \tag{67}$$

to partition $e^{\frac{t}{K}\frac{2U}{9}[\hat{x}_i^3,\hat{p}_i^3]}$ into terms involving $e^{i\left(\frac{t}{K}\frac{2U}{9}\right)^{1/2}\hat{x}_i^3}$ and $\mathcal{F}_i e^{i\left(\frac{t}{K}\frac{2U}{9}\right)^{1/2}\hat{x}_i^3}\mathcal{F}_i^\dagger$. Note that in Eq. (67), τ is proportional to $(t/K)^{1/2}$, thus the error is proportional to $(t/K)^2$. The expanded form of the time-evolution operator is given by

$$
\begin{aligned}
e^{itH} = \Bigg(\prod_{\{i,j\}:i<j} & e^{-i\frac{t}{K}J\hat{x}_i\hat{x}_j}\mathcal{F}_i\mathcal{F}_j e^{-i\frac{t}{K}J\hat{x}_i\hat{x}_j}\mathcal{F}_j^\dagger\mathcal{F}_i^\dagger \prod_i e^{i\frac{t}{K}\frac{U}{2}\hat{x}_i^4}\mathcal{F}_i e^{i\left(\frac{t}{K}\frac{2U}{9}\right)^{1/2}\hat{x}_i^3}\mathcal{F}_i^\dagger e^{i\left(\frac{t}{K}\frac{2U}{9}\right)^{1/2}\hat{x}_i^3}\mathcal{F}_i e^{-i\left(\frac{t}{K}\frac{2U}{9}\right)^{1/2}\hat{x}_i^3} \\
& \mathcal{F}_i^\dagger e^{-i\left(\frac{t}{K}\frac{2U}{9}\right)^{1/2}\hat{x}_i^3}\mathcal{F}_i e^{i\left(\frac{t}{K}\frac{2U}{9}\right)^{1/2}\hat{x}_i^3}\mathcal{F}_i^\dagger e^{i\left(\frac{t}{K}\frac{2U}{9}\right)^{1/2}\hat{x}_i^3}\mathcal{F}_i e^{-i\left(\frac{t}{K}\frac{2U}{9}\right)^{1/2}\hat{x}_i^3}\mathcal{F}_i^\dagger \\
& e^{-i\left(\frac{t}{K}\frac{2U}{9}\right)^{1/2}\hat{x}_i^3}\mathcal{F}_i e^{i\frac{t}{K}\frac{U}{2}\hat{x}_i^4}\mathcal{F}_i^\dagger e^{-i\frac{t}{K}U\hat{x}_i^2}\mathcal{F}_i e^{-i\frac{t}{K}U\hat{x}_i^2}\mathcal{F}_i^\dagger \Bigg)^K + O(\mathcal{R}). \quad (68)
\end{aligned}
$$

The error term that arises from Eq. (67) accumulates K times, thus the contribution to the error in the final expression is proportional to $K \cdot \frac{t^2}{K^2} = \frac{t^2}{K}$ and can be absorbed into the existing error term. The quartic term in the expansion, $e^{i\frac{t}{K}\frac{U}{2}\hat{x}_i^4}$, can be decomposed exactly as shown in Eq. (35).

3.1.1 Dipole Interaction

The Bose-Hubbard Hamiltonian can be extended in the case where dipolar bosons are trapped in the optical lattice. An external electric field can be applied to polarize the particles in a certain orientation, and the resulting system will have dipolar interactions leading to additional terms in the Hamiltonian. Truncating these interactions to the dominating term adds a dipole-dipole nearest neighbor interaction

[36], that is given by

$$H_{nn} = V_{\text{dip}} \sum_{\{i,j\}:i<j} \hat{n}_i \hat{n}_j. \tag{69}$$

Other nearest neighbor terms arising from this interaction represent one-particle tunneling and pair tunneling, but in this work I will focus on the dominant V_{dip} as above.

Following the procedure from before, we can expand the Hamiltonian in terms of $\hat{p}$ and $\hat{x}$ operators, then rotate the $\hat{p}$s into $\hat{x}$s, and decompose into gates from the universal set. Again to error $O(N^2t^2/K)$, the sequence of gates includes the sequence of four Gaussian gates given by

$$e^{-i\frac{t}{K}\frac{V_{\text{dip}}}{2}\hat{x}_i^2}\mathcal{F}_i e^{-i\frac{t}{K}\frac{V_{\text{dip}}}{2}\hat{x}_i^2}\mathcal{F}_i^\dagger e^{-i\frac{t}{K}\frac{V_{\text{dip}}}{2}\hat{x}_j^2}\mathcal{F}_j e^{-i\frac{t}{K}\frac{V_{\text{dip}}}{2}\hat{x}_j^2}\mathcal{F}_j^\dagger, \tag{70}$$

and four quartic terms given by

$$e^{i\frac{t}{K}V_{\text{dip}}\hat{x}_i^2\hat{x}_j^2}\mathcal{F}_j e^{i\frac{t}{K}V_{\text{dip}}\hat{x}_i^2\hat{x}_j^2}\mathcal{F}_j^\dagger \mathcal{F}_i e^{i\frac{t}{K}V_{\text{dip}}\hat{x}_i^2\hat{x}_j^2}\mathcal{F}_i^\dagger \mathcal{F}_i \mathcal{F}_j e^{i\frac{t}{K}V_{\text{dip}}\hat{x}_i^2\hat{x}_j^2}\mathcal{F}_j^\dagger \mathcal{F}_i^\dagger. \tag{71}$$

Each of these two-mode quartic operators involving $\hat{x}_i^2\hat{x}_j^2$ can be decomposed exactly as shown in Eq. (39). This leads to the following relation

$$e^{i\frac{t}{K}V_{\text{dip}}\hat{x}_i^2\hat{x}_j^2} = \mathcal{F}_i e^{i2\hat{x}_i\hat{x}_j}\mathcal{F}_i^\dagger e^{i\frac{t}{K}\frac{V_{\text{dip}}}{12}\hat{x}_i^4}\mathcal{F}_i e^{-i4\hat{x}_i\hat{x}_j}\mathcal{F}_i^\dagger e^{i\frac{t}{K}\frac{V_{\text{dip}}}{12}\hat{x}_i^4}\mathcal{F}_i e^{i2\hat{x}_i\hat{x}_j}\mathcal{F}_i^\dagger e^{-i\frac{t}{K}\frac{V_{\text{dip}}}{6}\hat{x}_i^4} e^{-i\frac{t}{K}\frac{V_{\text{dip}}}{6}\hat{x}_j^4}, \tag{72}$$

where again the single-mode quartic operations can be decomposed as in Eq. (35).

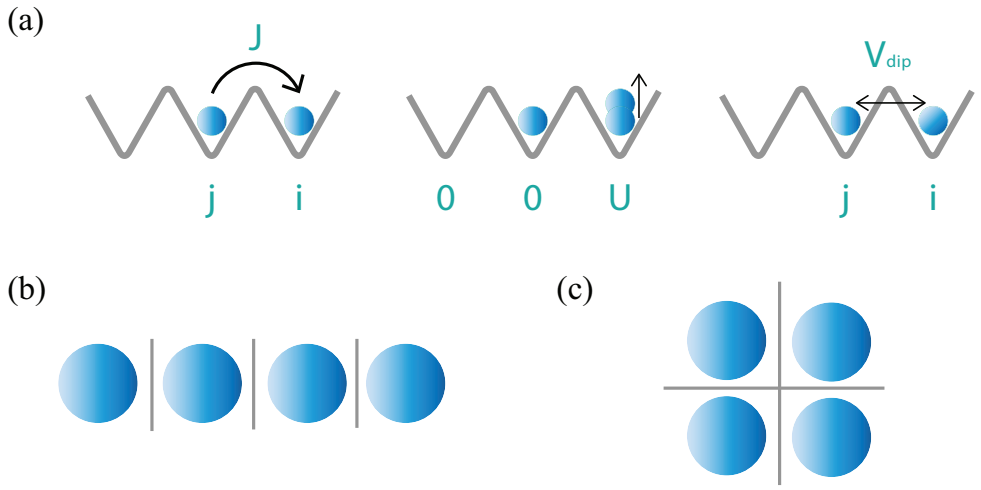
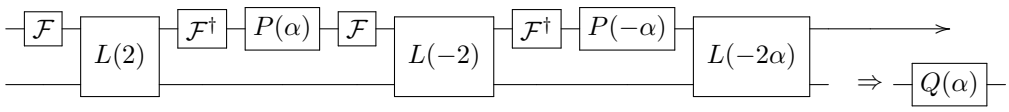


Figure 1: (a) A visualization of the effects of the terms in the Bose-Hubbard Hamiltonian. Here, J is the tunneling coefficient which dictates the movement of particles from one site to a neighboring site, U is the on-site interaction between two particles, and V_{dip} is the leading term of a dipole interaction between particles in neighboring sites. Also shown are two simple examples of lattices for which the circuit implementations are examined, (b) a one-dimensional four-node lattice and (c) a two-dimensional four-node lattice.

3.2 Circuit Implementations and Gate Counts

In this section, I show the quantum circuits implementing the time evolution of the Bose-Hubbard model. I start by examining the circuit for a one-dimensional four-node lattice, and then examine the additional circuit of the dipole interaction term. This is then generalized to two-dimensional lattices of size $n \times n$.

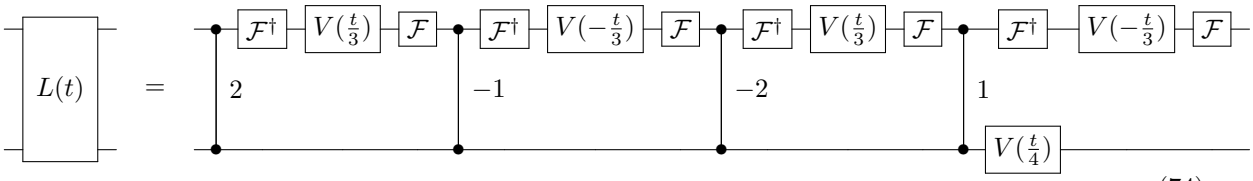
The notation used in the circuits follows from the notations introduced in Eq. (10) and Eq. (11). Note that the quartic gates in each circuit, denoted by $Q(t)$, can be decomposed in terms of gates from the universal set as in Eq. (35). This decomposition utilizes multiple modes and is demonstrated by the circuit



(73)

Acting upon the two wires with this sequence of gates is equivalent to acting on the bottom wire by the desired quartic gate. The top wire acts as an ancillary mode used in the decomposition and can be re-purposed afterwards [4]. The L gate is a multimode sequence of gates as in Eq. (29) with circuit given

by

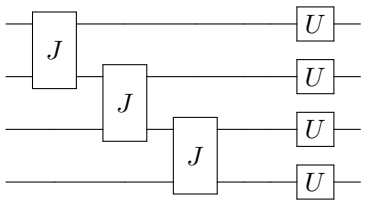


(74)

In total one quartic gate contributes 28 Fourier transform gates, 2 quadratic gates, 15 cubic gates, and 12 Cz gates.

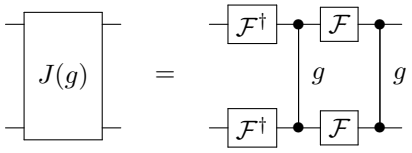
3.2.1 1-D Lattice Circuits

To present an example circuit for a single time step as in Eq. (68), we consider a 1-D lattice with four nodes as in Fig. 1(b). The circuit is given by



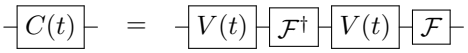
(75)

Here, the gate J is given by



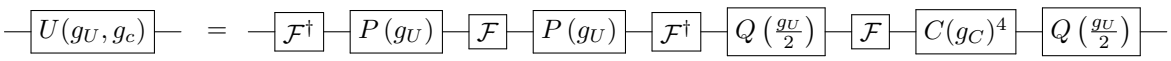
(76)

The Cz gate is performed in between each pair of Fourier transform gates and g is taken to be $g = tJ/K =: g_J$. To simplify the U gate we introduce a series of cubic and Fourier transform gates notated by C, given by the circuit



(77)

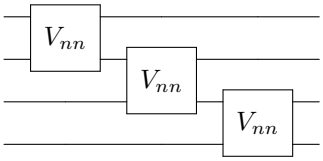
The U gate is then given by the circuit, with $g_U = \frac{tU}{K}$ and $g_C = (\frac{t}{K}\frac{2U}{9})^{1/2}$,



(78)

The gates from the universal set needed for this circuit will be denoted in the form $(\mathcal{F}, P, V, \mathrm{Cz})$. In the present case, we have $(\mathcal{F}, P, V, \mathrm{Cz}) = (284, 24, 152, 102)$. Thus, for one time step, we need 284 Fourier gates, 24 quadratic gates (squeezers and rotations), 152 cubic gates, and 102 Cz gates, with the given gate times g, g_U, and g_C.

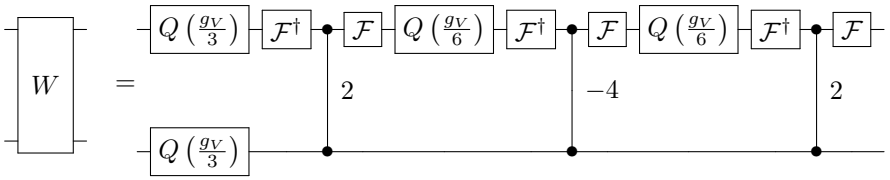
The additional dipole term may also be implemented in a circuit for a single time step in a 1-D lattice of 4 nodes. This circuit is given by



(79)

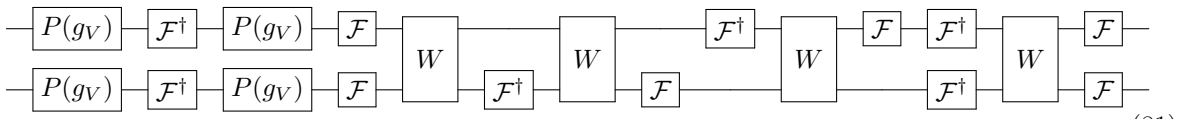
To expand the V_{nn} gate, the decomposition of the two-mode quartic gate in Eq. (39) can be denoted by

W, which has the circuit



(80)

Here, $g_V = tV_{\text{dip}}/2K$. The V_{nn} gate is then given by

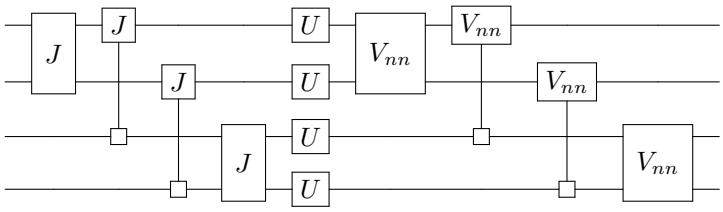


(81)

Using a similar gate count notation as before, the dipole part of the circuit for the 1-D lattice will have a gate count of $(\mathcal{F}, P, V, \text{Cz}) = (1452, 108, 720, 612)$. This means the total circuit including all of the U and J terms will have a gate count of $(\mathcal{F}, P, V, \text{Cz}) = (1736, 132, 872, 714)$ for a single time step.

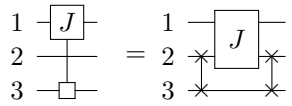
3.2.2 2-D Lattice Circuits

In this section, two-dimensional lattices of size $n \times n$ are examined. First, consider a 2×2 lattice with four total nodes as in Fig. 1(c). The circuit has the form



(82)

Here, a new notation has been introduced for the two-mode J and V_{nn} gates over two non-neighboring wires. This can be implemented on a circuit with only nearest neighbor coupling by swapping neighboring modes, applying the J or V_{nn} gates and then swapping back. For example,



(83)

Note that the square box on the circuit indicates the other qumode that is being acted upon, and the wires connected by crosses denote a swap operation between two modes which can be performed with a beamsplitter. This can similarly be done for an $n \times n$ lattice where, if the Bose-Hubbard model has nearest neighbor couplings, at most n swaps are needed on either side of a gate. For an $n \times n$ lattice the first part of the circuit, which is the nearest neighbor pattern involving the J gates, is given in Fig. 2.

The final gate count for the $n \times n$ lattice can now be totaled. Following the notation as before, we also include a count for the number of swaps needed. For each J gate the count is $(\mathcal{F}, \mathrm{Cz}) = (4, 2)$, and for the $n \times n$ lattice there are $2(n^2 - n)$ J gates and $2(n^3 - n^2)$ swaps, which gives us a gate count of $(\mathcal{F}, \mathrm{Cz}, \mathrm{SWAP}) = \big(8(\mathrm{n}^2 - \mathrm{n}), 4(\mathrm{n}^2 - \mathrm{n}), 2(\mathrm{n}^3 - \mathrm{n}^2)\big)$. As shown above, each U gate has a count of $(\mathcal{F}, P, V, \mathrm{Cz}) = (68, 6, 38, 0)$ and in the lattice we have n^2 of them, giving a total count for the U gates of $(\mathcal{F}, P, V, \mathrm{Cz}) = (68\mathrm{n}^2, 6\mathrm{n}^2, 38\mathrm{n}^2, 0)$. Finally, each V_{nn} gate has a count of $(\mathcal{F}, P, V, \mathrm{Cz}) = (484, 36, 240, 204)$, and in the lattice the V_{nn} gates follow the same pattern as the J gates, so we have a total contribution from the V_{nn} gates of $(\mathcal{F}, P, V, \mathrm{Cz}, \mathrm{SWAP}) = \big(968(n^2 - n), 72(n^2 - n), 480(n^2 - n), 408(n^2 - n), 2(n^3 - n^2)\big)$.

Therefore, the final gate count for our $n \times n$ lattice is

$$(\mathcal{F}, P, V, \mathrm{Cz}, \mathrm{SWAP}) = \big(1044n^2 - 976n, 78n^2 - 72n, 518n^2 - 480n, 412(n^2 - n), 4(n^3 - n^2)\big). \tag{84}$$

Note that this is the gate count for each time step of length t/K in the series of gates simulating e^{iHt}, as in Eq. (68) and Eqs. (70) to (72).

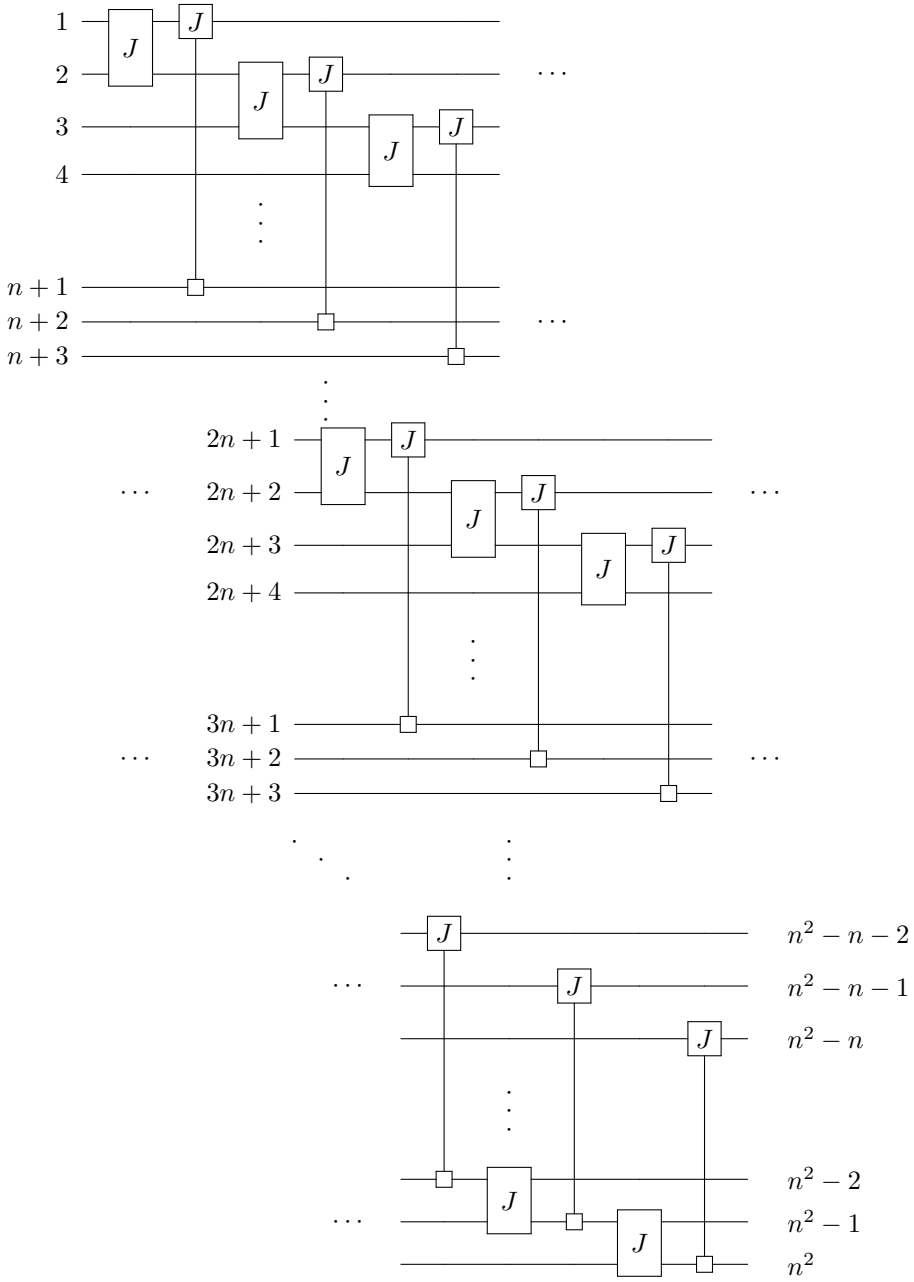
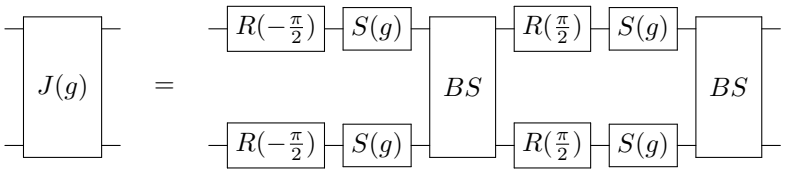


Figure 2: Circuit diagram for the J terms of the Bose-Hubbard Hamiltonian in Eq. (60) applied to a two-dimensional, $n \times n$ lattice. The dipole interaction term as in Eq. (69) will also have the same pattern, but will have gates notated with V_{nn}

3.3 Implementation and Errors

Note that, as discussed in Section 1.3, the Gaussian elements of the circuits outlined in this chapter can be implemented deterministically with linear optics whereas the higher-order gates are more complex and contain probabilistic elements.

Examining the J gate as in Eq. (76). This circuit element consists of Fourier transforms and Cz gates which are single-mode Gaussian and multi-mode Gaussian operations and as such can be implemented with linear optics. For the J gate, the Fourier transforms are implemented simply with rotations of $\frac{\pi}{2}$, whereas the Cz gates require squeezers and the multi-mode transformation of beamsplitters. More precisely, the J gate can be optically implemented in the following way



(85)

where squeezing operations are denoted by S, beamsplitters by BS, and rotations by R.

In order to implement higher-order gates we require more than the set of linear optical elements. The cubic phase operators denoted by the $V(t)$ gates in the above circuits, are an example of these higher-order operations. To implement the cubic phase gate we add to the set of optical elements a photon counting measurement, which introduces the non-linearity needed. The process for creating a

cubic phase operation in terms of these elements is discussed in Section 1.3, and the circuit is shown in Eq. (23). Note that in the case where a Kerr interaction is available, given by $e^{it\hat{n}_i^2}$, it may be added to the universal set and used to directly implement the non-linear parts of the U gates [3, 64, 65].

When performing the gate decomposition and analyzing the makeup of example circuits, note that all gate counts are given for a single Trotter time step. Let the desired accuracy of simulating e^{itH} be given by ϵ. The accuracy is dependent on the choice of number of time slices K, the total simulation time t, and the number of sites N. From Eq. (64), we can determine K to achieve a given accuracy. Such a K is given by

$$K = O\left(\frac{N^2 t^2}{\epsilon}\right). \tag{86}$$

The commutator simulation from Eq. (67) contributes at most in the same order as the sum formula Eq. (64). Our final product of operations for the Bose-Hubbard Hamiltonian is raised to the power of K, therefore we must repeat each circuit presented in this work K times in order to get the desired error of ϵ.

Another important source of error is the effect of finite squeezing. As discussed in Section 1.3, the optical implementation of the gates in each circuit will require the use of squeezing. In any experimental setup the squeezing will be finite and the end result with be dependent on a squeezing factor s [20, 21]. For example, consider an optical implementation of the cubic phase gate where a photon counting measurement is made on a displaced two-mode squeezed state. To construct the two-mode squeezed state, two squeezed states,

which ideally are zero-momentum eigenstates, are entangled. However, realistically the quadratures can only be finitely squeezed, in for example the momentum quadrature

$$|0\rangle_p \to \int dp\ e^{-(p)^2/(2s)}\,|p\rangle\,. \tag{87}$$

The cubic phase gate is then modulated by a Gaussian envelope with zero mean and variance that depends on the squeezing factor s [20]. The result of this is a distortion effect which is inversely proportional to the amount of squeezing applied.

4 Other Applications

In this Chapter I examine two algorithms which were designed for photonic quantum computers. Both of the algorithms require the implementation of an operator in the form $e^{it\hat{H}}$, which is covered by the exact decomposition method described in Chapter 2. As such, the implementation of these algorithms may benefit greatly from the use of exact decompositions. Table 2 also compiles a list of other quantum algorithms that may benefit from exact gate decompositions.

The first algorithm describes a method for solving non-homogeneous partial differential equations by adapting a mathematical method to efficiently invert differential operators [7]. The subsection describing this algorithm as well as its decomposition follows from the publication, for which I was a co-author. The second is an algorithm which performs Monte Carlo evaluation of multi-dimensional integrals in the continuous-variable setting [37].

4.1 Quantum Algorithm for Non-Homogeneous Linear Partial Differential Equations

It has been shown in various articles how quantum computers can excel at solving systems of linear equations [66–70]. In these examples, given a sparse $N \times N$ matrix A and a vector $\mathbf{b} = (b_1, \ldots, b_N)$, the goal is to find a vector $\mathbf{x} = (x_1, \ldots, x_N)$ satisfying the equation $A\mathbf{x} = \mathbf{b}$. These quantum algorithms take

as input the quantum state $|\mathbf{b}\rangle = \sum_{i=1}^{N} b_i |i\rangle$ and efficiently perform matrix inversion to prepare the state $|\mathbf{x}\rangle = A^{-1} |\mathbf{b}\rangle$ encoding the solution of the linear system of equations.

In the CV version of this problem the inputs are a function $f(\mathbf{x})$ over $\mathbb{R}^N$ as well as a differential operator A. In general, A is expressed as a function of the variables and their partial derivatives: $A = A(x_1, \ldots, x_N, \frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_N})$. The goal of the algorithm in this case is to find a function $\psi(\mathbf{x})$ satisfying the linear partial differential equation $A\psi(\mathbf{x}) = f(\mathbf{x})$. This differential equation is non-homogeneous whenever $f(\mathbf{x}) \neq 0$. Similar to the quantum algorithms for solving a linear system of equations, the non-homogeneous differential equation problem can be solved by first finding the inverse operator A^{-1} and then using it to compute the function $\psi(\mathbf{x}) = A^{-1} f(\mathbf{x})$. Note that on a physical quantum computer the full wavefuntion $\psi(\mathbf{x})$ cannot be accessed, only measurement outcomes on the resulting output state $|\Psi\rangle$.

In order to find the inverse operator A^{-1} a Fourier decomposition technique from Ref. [69] is used:

$$\hat{A}^{-1} = \frac{i}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dx dy \, \Theta(x) \, y e^{-y^2/2} e^{-i\hat{A}xy}, \tag{88}$$

where $\Theta(x)$ is the heaviside step function, which is approximated by a step function quantum state with finite width. The resulting output state is given by

$$|\Psi\rangle = \frac{i}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dx dy \, \Theta(x) \, y e^{-y^2/2} e^{-i\hat{A}xy} |f\rangle |x\rangle |y\rangle , \tag{89}$$

which is equivalent to having applied the operation $e^{-i\hat{A}\hat{X}\hat{Y}}$ to the three modes, $|f\rangle$, $|x\rangle$, and $|y\rangle$. Here, $\hat{X}$ and $\hat{Y}$ are position operators acting on their respective modes. Performing a measurement and post-selecting on observing $p = 0$ on the x and y modes results in having acted with $\hat{A}^{-1}$ on state $|f\rangle$. In order to implement the algorithm a suitable operator $\hat{A}$ must be chosen and the operator $e^{-i\hat{A}\hat{X}\hat{Y}}$ must be decomposed in terms of operators contained in the universal set.

As shown in Eq. (5), the momentum operator $\hat{p}$ acting on any arbitrary position space wavefuntion will act as a partial derivative (note that $\hat{p} \equiv \hat{\vec{p}}$ as it is defined in Eq. (5)). Therefore choosing a general operator $\hat{A}$ that includes a linear combination of $\hat{p}$ and $\hat{p}^2$ will encompass a large class of differential operators. These include Poisson's equation, the heat equation, and the wave equation. More specifically let $\hat{A}$ have the form

$$\hat{A} = \lambda \mathbb{1} + \sum_{j=1}^{N} a_j \hat{x}_j + b_j \hat{p}_j + \alpha_j \hat{x}_j^2 + \beta_j \hat{p}_j^2, \tag{90}$$

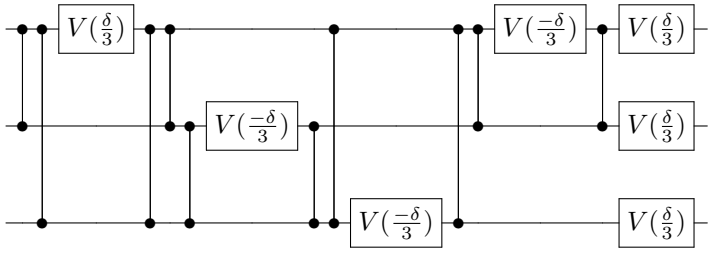
where $\lambda, a_j, b_j, \alpha_j$, and β_j are real constants. After performing a Trotter-Suzuki decomposition (Eq. (13)) on $e^{-i\hat{A}\hat{x}_k\hat{x}_l}$ the operators left to be decomposed will have the forms $e^{it\hat{x}_j\hat{x}_k}$, $e^{it\hat{x}_j\hat{x}_k\hat{x}_l}$ or $e^{it\hat{x}_j^2\hat{x}_k\hat{x}_l}$, up to Fourier transform, where the subindices denote which mode the operators act on. The operator $e^{it\hat{x}_j\hat{x}_k}$ is already an element of the chosen universal set in Eq. (7) so it remains to decompose the second two operations. The process for decomposing $e^{it\hat{x}_j\hat{x}_k\hat{x}_l}$ is shown in Eq. (30 - 32) in Chapter 2, and will result

in the following decomposition

$$e^{i2\delta\hat{x}_j\hat{x}_k\hat{x}_l} = e^{i2\hat{p}_j\hat{x}_k}e^{i2\hat{p}_j\hat{x}_l}e^{\frac{i\delta}{3}\hat{x}_j^3}e^{-i2\hat{p}_j\hat{x}_l}e^{-i2\hat{p}_j\hat{x}_k}e^{i2\hat{p}_k\hat{x}_l}e^{\frac{-i\delta}{3}\hat{x}_k^3}e^{-i2\hat{p}_k\hat{x}_l}$$

$$e^{i2\hat{p}_l\hat{x}_j}e^{\frac{-i\delta}{3}\hat{x}_l^3}e^{-i2\hat{p}_l\hat{x}_j}e^{i2\hat{p}_j\hat{x}_k}e^{\frac{-i\delta}{3}\hat{x}_j^3}e^{-i2\hat{p}_j\hat{x}_k}e^{\frac{i\delta}{3}\hat{x}_j^3}e^{\frac{i\delta}{3}\hat{x}_k^3}e^{\frac{i\delta}{3}\hat{x}_l^3}. \tag{91}$$

This decomposition can also be expressed as the circuit



(92)

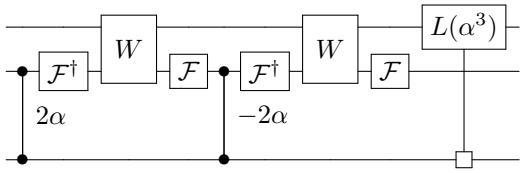
where each Cz gate strength parameter is either 2 or -2, and also each of the Fourier transform gates that would surround the Cz gates have been neglected for the purpose of readability.

The decomposition for the gate $e^{it\hat{x}_j^2\hat{x}_k\hat{x}_l}$ follows from Eq. (38) with $n = 2$, and is given by

$$e^{2i\alpha^2\hat{x}_j^2\hat{p}_k\hat{p}_l} = e^{2i\alpha\hat{x}_k\hat{x}_l}e^{-i\alpha\hat{x}_j^2\hat{p}_k^2}e^{-2i\alpha\hat{x}_k\hat{x}_l}e^{i\alpha\hat{x}_j^2\hat{p}_k^2}e^{i\alpha^3\hat{x}_j^2\hat{p}_l}. \tag{93}$$

where the decompositions, up to Fourier transform for $e^{-i\alpha\hat{x}_j^2\hat{p}_k^2}$ and $e^{i\alpha^3\hat{x}_j^2\hat{p}_l}$ are given in Eq. (39) and

Eq. (29). The circuit for this gate is given by:



(94)

where the circuits for the W and L gates are as in Eq. (80) and Eq. (74). Neglecting Fourier transform gates which are very inexpensive to implement, the exact decompositions require 17 and 249 gates from the universal set respectively, whereas the standard commutator approximation method [3] would require about 10^8 and 10^9 gates for a precision of 10^{-3}.

Figures 3 and 4 show the outputs of a simulation of the algorithm for solving Poisson's equation. A charge distribution corresponding to the wavefunction of the input state is given to the algorithm which then calculates the electric potential. Since the simulation is done on a classical computer this involves calculating the integral of the corresponding inverse operator dictated by the algorithm to find a function corresponding to the electric potential. The electric field can then be calculated from the gradiant of the potential function. The electric field lines are plotted in each figure. Note that the simulation may run with any input function for the charge distribution but on a physical quantum computer the inputs must be states that can be prepared in advance. Also, on a quantum computer the output state of the algorithm must be repeatedly measured to reveal areas of large electrostatic potential. Figures 3 and 4 demonstrate two input charge distributions for which the input states are not difficult to prepare.

This quantum algorithm has a polynomial run time in the dimension of A, which is an exponential improvement over classical algorithms which compute full solutions to partial differential equations.

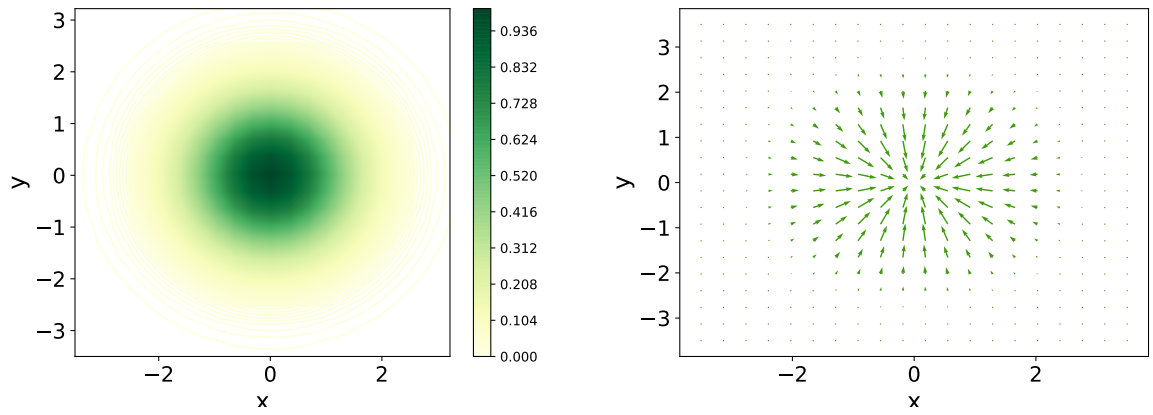


Figure 3: (Left) Charge distribution $\rho(x, y) = e^{-\frac{x^2}{2}} e^{-\frac{y^2}{2}}$ which is equivalent, up to normalization, to the wavefunction of a two-mode Gaussian input state. (Right) Electric field lines reconstructed from the output state of the quantum algorithm.

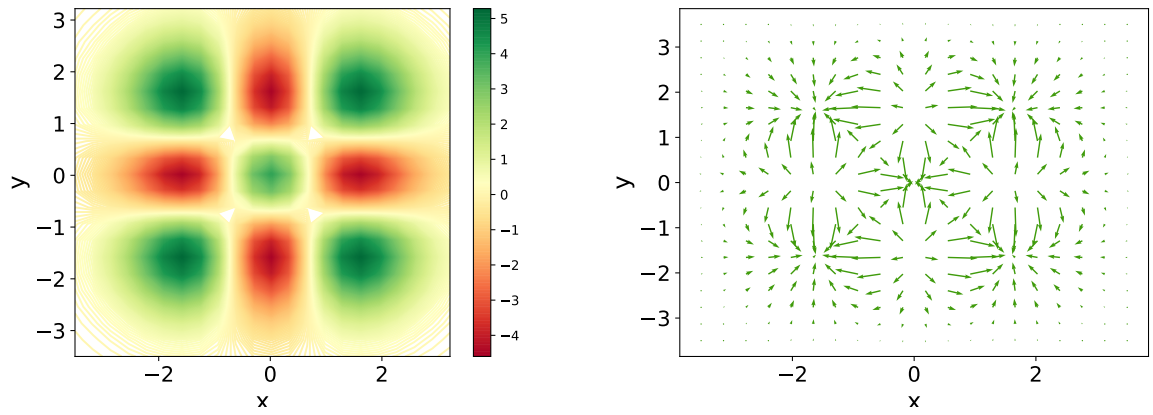


Figure 4: (Left) Charge distribution $\rho(x, y) = (4x^2 - 2)(4y^2 - 2)e^{-\frac{x^2}{2}}e^{-\frac{y^2}{2}}$ which is equivalent, up to normalization, to the wavefunction of the two-mode input state $|f\rangle = |2\rangle\,|2\rangle$ with two photons in each state. Green quadrants are regions of positive charge and red are regions of negative charge. (Right) Electric field lines reconstructed from the output state of the quantum algorithm.

4.2 Photonic Quantum Algorithm for Monte Carlo Integration

Monte Carlo methods are computational methods which rely on repeated random sampling to provide numerical solutions to a problem. These methods have a broad use in many applications such as finance, machine learning, database search, optimization, and sampling [71–74]. Quantum algorithms for Monte-Carlo methods have been described in both the qubit setting [75, 76] and continuous-variable setting [37, 77].

More specifically Ref. [37] presents a continuous-variable quantum algorithm which performs Monte-Carlo integrations. This algorithm provides a solution to the integral

$$I = \int_{\mathbb{R}^n} d\vec{x}\, p(\vec{x}) f(\vec{x}), \tag{95}$$

where $f(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ is a real function, bounded as $0 < f(\vec{x}) \leq 1$ for all $\vec{x}$, which represents a random variable of outcomes distributed by a multidimensional probability distribution $p(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$.

The algorithm requires the following steps on four modes and imprints the final result proportional to I in the final mode:

1. Prepare a state according to the probability distribution $p(\vec{x})$ in the first mode. This is done by

applying a unitary operation to a vaccum state:

$$G\left|vac\right\rangle = \int dx \sqrt{p(x)}\left|x\right\rangle_1 \tag{96}$$

For example, for a Gaussian probability density the unitary operation G can be implemented with linear optics. In other cases G may need to be decomposed into operations from a universal set, using either approximate methods or if possible with an exact decomposition from Chapter 2.

2. The random variable function $f(x)$ is imprinted by applying a three mode operator

$$H = e^{-i\left(1/\sqrt{f(\hat{x}_1)}\right)\hat{p}_2\hat{p}_3}, \tag{97}$$

on the mode that has been prepared with G as well as two other squeezed ancillary modes. In order to implement this operator, the function $1/\sqrt{f(\hat{x}_1)}$ is approximated by a polynomial function $h(\hat{x}_1)$ which is a polynomial in $\hat{x}_1$. Note that as $f(\hat{x}_1) \to 0$ the approximation $h(\hat{x}_1) \to \infty$ and as a result it may no longer be possible to find a good approximation which the algorithm relies on for implementation. In this case even if a good approximation can be found it may also no longer be practical to implement such a large polynomial in $\hat{x}_1$. I will assume for practicality that we are then sufficiently far from this region. The resulting operation H can be decomposed with the general equation (38) after splitting each of the terms in the polynomial with the Trotter-Suzuki

approximation as in Eq. (13).

3. The algorithm then provides a speedup over its classical analog through the use of a CV amplitude estimation. This involves a phase estimation operator that uses an additional fourth resource mode denoted with ϕ, and is given by

$$Q = e^{-i\left(1/\sqrt{f(\hat{x}_1)}\right)\hat{p}_2\hat{p}_3\hat{p}_\phi} = e^{-ih(\hat{x}_1)\hat{p}_2\hat{p}_3\hat{p}_\phi}, \tag{98}$$

where again the function $1/\sqrt{f(\hat{x}_1)}$ is approximated with a polynomial $h(\hat{x}_1)$. This operator again depends on the choice of $h(\hat{x}_1)$, but regardless of this choice the operation will be covered by the exact decomposition method as detailed in Section 2.2.1. Assuming $h(\hat{x}_1) = \hat{x}_1^n$, then because there are four total modes, $Nn = 4n$ is always even and therefore the single mode operation $e^{it\hat{x}_1^{4n}}$ can be decomposed exactly with Eq.(41). Also, since the final three modes are all to unit power, any one of them may be used as the exponent of the central operator in unitary conjugation. Therefore both of the restrictions of this portion of the method have been met regardless of n. By linearity, the same holds for a general polynomial $h(\hat{x}_1) = \sum_n a_n \hat{x}_1^n$. Finally, postselecting on the resource modes gives a success probability which is proportional to the desired integral I.

Table 3 shows the gates counts of the operations in step 2. and 3. for $h(\hat{x}_1)$ a polynomial of order five or less. If $h(\hat{x}_1)$ is a more general polynomial with multiple terms, then Trotter-Suzuki approximation can be used to split the operation into the terms shown on the table. At higher order these gate counts

do not surpass those of the commutator approximation method for the first row (as shown in Table 1). Although, they do seem to scale poorly as the power of the first mode is increased. If a higher order polynomial $h(\hat{x})$ is needed then the gate counts may become intractable both with an exact decomposition and with the commutator approximation.

This quantum algorithm claims a potential quadratic speedup in estimating integrals on a CV quantum computer. Although, note that because the function $f(\vec{x})$ is bounded as $0 < f(\vec{x}) \leq 1$, then the integral I that is returned by the algorithm is also bounded by $I \leq 1$.

$h(\hat{x}_1)$	Target gate	Exact decomposition gate count
$\hat{x}_1$	$e^{it\hat{x}_1\hat{p}_2\hat{p}_3\hat{p}_\phi}$ $e^{it\hat{x}_1\hat{p}_2\hat{p}_3}$	440 gates 17 gates
$\hat{x}_1^2$	$e^{it\hat{x}_1^2\hat{p}_2\hat{p}_3\hat{p}_\phi}$ $e^{it\hat{x}_1^2\hat{p}_2\hat{p}_3}$	3749 gates 249 gates
$\hat{x}_1^3$	$e^{it\hat{x}_1^3\hat{p}_2\hat{p}_3\hat{p}_\phi}$ $e^{it\hat{x}_1^3\hat{p}_2\hat{p}_3}$	47061 gates 1337 gates
$\hat{x}_1^4$	$e^{it\hat{x}_1^4\hat{p}_2\hat{p}_3\hat{p}_\phi}$ $e^{it\hat{x}_1^4\hat{p}_2\hat{p}_3}$	5.6×10^5 gates 4462 gates
$\hat{x}_1^5$	$e^{it\hat{x}_1^5\hat{p}_2\hat{p}_3\hat{p}_\phi}$ $e^{it\hat{x}_1^5\hat{p}_2\hat{p}_3}$	7.0×10^6 gates 15289 gates

Table 3: Exact decomposition gate counts for the unitary operators needed in the quantum Monte-Carlo integration algorithm. The upper gate in each row corresponds to the controlled unitary needed to perform amplitude estimation, and the lower unitary imprints the function $f(\vec{x})$ which represents a random variable of outcomes distributed by $p(\vec{x})$ needed to approximate the desired integral.

5 Discussion and Conclusions

In this work I have detailed a systematic method for performing exact gate decompositions on a photonic quantum computer. In essence, the method works by expressing a target Hamiltonian as a linear combination of polynomials, then finding exact decompositions of these polynomials using unitary conjugation in combination with the lemma to Baker-Campbell-Hausdorff. The unitary operations covered by this method are a large set of operations arising in photonic quantum algorithms and the simulation of bosonic systems. Compared to previous techniques such as the standard commutator approximation, these methods can yield reductions in gate count of several orders of magnitude, with the added advantage that the target unitaries are decomposed exactly. Chapter 2 provides a detailed description of the method as well as comparisons of gate counts with the standard commutator approximation.

Despite its wide applicability, the method does not produce exact decompositions for all possible bosonic gates. Notably, Hamiltonians that contain products of both $\hat{x}$ and $\hat{p}$ quadrature operators – for instance operators of the form $\hat{H} = \hat{x}^n\hat{p}^m + \hat{p}^m\hat{x}^n$ – are not covered by the method. Operators of the form ($\hat{H} = \hat{x}^n\hat{p}^m + \hat{p}^m\hat{x}^n$) are challenging because the method relies on commutation relations to form polynomials in $\hat{x}$. Having a mixture of $\hat{x}$ and $\hat{p}$ operators acting on the same mode in the desired gate may mean the commutators no longer simplify or terminate to zero in the Baker-Campbell-Hausdorff (28) or Zassenhaus formula (12). I have found that this can lead to systems of equations which are no longer linear, unlike the systems dealt with in Section 2.2.1. An outstanding open question resulting from this

work is to fully characterize the set of operations that can be decomposed exactly using the techniques presented here, using the chosen universal set or otherwise. As well, given any arbitrary operation it is not clear whether an exact decomposition may exist in general.

As the decomposition method presented here applies to photonic quantum computers, a natural application would be to examine a system of bosonic particles. In Chapter 3, I presented a thorough examination of the Bose-Hubbard Hamiltonian, which describes a system of bosonic particles trapped in an optical lattice. Using the exact decomposition methods as in Chapter 2, as well as approximate methods, I provided circuit diagrams and gate counts for the implementation of the time-evolution of the Bose-Hubbard Hamiltonian. The circuits discussed include a simple four-node, one-dimensional optical lattice for the Bose-Hubbard model and general two-dimensional lattices of size $n \times n$. The final gate count for a $n \times n$ lattice is given in Eq. (84) in terms of the number of gates of each type needed from the universal set. The procedure used in this case can be extended to other similar Hamiltonians. An efficiently simulable subclass of the Bose-Hubbard Hamiltonian is the bosonic tight-binding Hamiltonian [78] with applications in condensed matter and solid state physics. The tight-binding Hamiltonian coupled to a bath of harmonic oscillators appears also in the study of exciton dynamics in photosynthetic complexes [79]. Simulating such systems can provide another application for continuous-variable photonic quantum processors.

In Chapter 4, I provided a description of two quantum algorithms designed for a photonic quantum computer. Namely a method for solving non-homogeneous partial differential equations by adapting a

mathematical method to efficiently invert differential operators [7], and an algorithm which performs Monte-Carlo evaluation of multi-dimensional integrals in the continuous-variable setting [37]. Both of these algorithms require the implementation of unitary operators of the form $e^{it\hat{H}}$ that can be decomposed with the exact decomposition methods in Chapter 2. The quantum algorithm for solving partial differential equations required two types of unitaries for the choice of operator $\hat{A}$ as in Eq. (90) with gate counts of 17 and 249 respectively. These unitaries allowed for first and second derivative to be expressed in the algorithm, and while higher-order derivatives would require more complicated unitaries, the form they would take would still be covered by the decomposition in Eq. (38). The gate counts for the required operators of the Monte-Carlo integration algorithm are given in Table 3. Note that these gate counts assume that a fifth order polynomial $h(\hat{x}_1)$ is sufficient in approximating the function $1/\sqrt{f(\hat{x}_1)}$. Although a higher-order polynomial would lead to operators that are still decomposable with the exact method, the decompositions seem to scale very poorly in this case.

Going forward the challenge remains to fully characterize which decompositions can be done exactly, and if new approximate methods will be discovered that scale better with the total order of the gates. For CV quantum computers it would be useful to completely characterize the commutator algebra to allow for a general method of creating higher order operations following the work from [14]. Experimentally the implementation of non-linear gates remains a problem and as such even small decompositions for lower order gates may not see use in the short term. For example $e^{it\hat{x}^4}$ was exactly decomposed into 29 gates, 15 of which were cubic phase gates. While this is substantially better then standard approximate

methods, it may still be some time before multiple cubic phase gates can be implemented reliably in a circuit. Implementing a large number of optical elements in a small space is another ongoing challenge for CV quantum computing, with the main setback in this case again being non-linear gates which may require some sort of refrigeration system to implement accurate detectors. Another problem is signal loss over large distances and the need for a quantum analogue of signal repeaters [32]. While these problems are not addressed in this work, the techniques for my exact decomposition method still hold under the addition of imperfect gates and external errors.

Appendix A: Proofs and Derivations

Proof of Eq. (37)

Here I prove the identity

$$e^{2i\alpha^2\hat{p}_k\hat{x}_j^N} = e^{2i\alpha\hat{x}_j^{N-2}\hat{x}_k}e^{-i\alpha\hat{x}_j^2\hat{p}_k^2}e^{-2i\alpha\hat{x}_j^{N-2}\hat{x}_k}e^{i\alpha\hat{x}_j^2\hat{p}_k^2}e^{i\alpha^3\hat{x}_j^{2(N-1)}}. \tag{99}$$

The middle three operators on the right-hand side can be expanded with unitary conjugation as

$$e^{-i\alpha\hat{x}_j^2\hat{p}_k^2}e^{-2i\alpha\hat{x}_j^{N-2}\hat{x}_k}e^{i\alpha\hat{x}_j^2\hat{p}_k^2} = e^{-2i\alpha\left(e^{-i\alpha\hat{x}_j^2\hat{p}_k^2}\hat{x}_j^{N-2}e^{i\alpha\hat{x}_j^2\hat{p}_k^2}\right)\left(e^{-i\alpha\hat{x}_j^2\hat{p}_k^2}\hat{x}_k e^{i\alpha\hat{x}_j^2\hat{p}_k^2}\right)}. \tag{100}$$

Then using the lemma to BCH, the two factors in the exponent can be simplified to get

$$e^{-i\alpha\hat{x}_j^2\hat{p}_k^2}\hat{x}_j^{N-2}e^{i\alpha\hat{x}_j^2\hat{p}_k^2} = \hat{x}_j^{N-2}, \tag{101}$$

and

$$e^{-i\alpha\hat{x}_j^2\hat{p}_k^2}\hat{x}_k e^{i\alpha\hat{x}_j^2\hat{p}_k^2} = \hat{x}_k - \alpha\hat{x}_j^2\hat{p}_k. \tag{102}$$

The resulting two terms in the exponent are then separated using the Zassenhaus formula of Eq. (12):

$$e^{-2i\alpha\hat{x}_j^{N-2}\left(\hat{x}_k-\alpha\hat{x}_j^2\hat{p}_k\right)} = e^{-2i\alpha\hat{x}_j^{N-2}\hat{x}_k}e^{2i\alpha^2\hat{x}_j^N\hat{p}_k}e^{-\frac{1}{2}[-2i\alpha\hat{x}_j^{N-2}\hat{p}_k,\, 2i\alpha^2\hat{x}_j^N\hat{p}_k]}$$

$$= e^{-2i\alpha\hat{x}_j^{N-2}\hat{x}_k}e^{2i\alpha^2\hat{x}_j^N\hat{p}_k}e^{-i\alpha^3\hat{x}_j^{2(N-1)}}. \tag{103}$$

The two outside operators, $e^{-2i\alpha\hat{x}_j^{N-2}\hat{x}_k}$ and $e^{-i\alpha^3\hat{x}_j^{2(N-1)}}$ then cancel with the remaining two operators in Eq. (37), leaving the desired operator $e^{2i\alpha^2\hat{x}_j^N\hat{p}_k}$.

Proof of Eq. (39)

Here I prove the following exact decomposition formula for the gate $e^{i\alpha\hat{x}_j^2\hat{x}_k^2}$:

$$e^{i\alpha\hat{x}_j^2\hat{x}_k^2} = e^{i2\hat{p}_j\hat{x}_k}e^{i\frac{\alpha}{12}\hat{x}_j^4}e^{-i4\hat{p}_j\hat{x}_k}e^{i\frac{\alpha}{12}\hat{x}_j^4}e^{i2\hat{p}_j\hat{x}_k}e^{-i\frac{\alpha}{6}\hat{x}_j^4}e^{-i\frac{\alpha}{6}\hat{x}_k^4}. \tag{104}$$

We begin by expressing the operator $\hat{x}_j^2\hat{x}_k^2$ as a linear combination of polynomials:

$$\hat{x}_j^2\hat{x}_k^2 = \tfrac{1}{12}\left(\hat{x}_j+\hat{x}_k\right)^4 + \tfrac{1}{12}\left(\hat{x}_j-\hat{x}_k\right)^4 - \tfrac{1}{6}\hat{x}_j^4 - \tfrac{1}{6}\hat{x}_k^4, \tag{105}$$

which leads to the identity

$$e^{i\alpha\hat{x}_j^2\hat{x}_k^2} = e^{i\frac{\alpha}{12}(\hat{x}_j+\hat{x}_k)^4}e^{i\frac{\alpha}{12}(\hat{x}_j-\hat{x}_k)^4}e^{-i\frac{\alpha}{6}\hat{x}_j^4}e^{-i\frac{\alpha}{6}\hat{x}_k^4}. \tag{106}$$

Finally, from unitary conjugation it holds that

$$e^{i2\hat{p}_j\hat{x}_k}e^{i\frac{\alpha}{12}\hat{x}_j^4}e^{-i4\hat{p}_j\hat{x}_k}e^{i\frac{\alpha}{12}\hat{x}_j^4}e^{i2\hat{p}_j\hat{x}_k} = e^{i\frac{\alpha}{12}(\hat{x}_j+\hat{x}_k)^4}e^{i\frac{\alpha}{12}(\hat{x}_j-\hat{x}_k)^4}, \tag{107}$$

which gives Eq. (104) when replaced in Eq. (106).

Proof of Eq. (41)

Here I show the recursive decomposition for single-mode gates $e^{i\alpha\hat{x}_k^N}$:

$$e^{i\alpha\hat{x}_k^N} = e^{2i\hat{p}_j\hat{x}_k^{N/2}}e^{i\alpha\hat{x}_j^2}e^{-2i\hat{p}_j\hat{x}_k^{N/2}}e^{-i\alpha\hat{x}_j^2}e^{-2i\alpha\hat{x}_j\hat{x}_k^{N/2}}. \tag{108}$$

As usual, begin by expressing the target operator as a linear combination of polynomials:

$$\hat{x}_k^N = \left(\hat{x}_j + \hat{x}_k^{N/2}\right)^2 - \hat{x}_j^2 - \hat{x}_j\hat{x}_k^{N/2} \tag{109}$$

which leads to the identity

$$e^{i\alpha\hat{x}_k^N} = e^{i\alpha\left(\hat{x}_j+\hat{x}_k^{N/2}\right)^2}e^{-i\alpha\hat{x}_j^2}e^{-2i\alpha\hat{x}_j\hat{x}_k^{N/2}}. \tag{110}$$

From unitary conjugation it holds that

$$e^{2i\hat{p}_j\hat{x}_k^{N/2}}e^{i\alpha\hat{x}_j^2}e^{-2i\hat{p}_j\hat{x}_k^{N/2}} = e^{i\alpha\left(\hat{x}_j+\hat{x}_k^{N/2}\right)^2}, \tag{111}$$

which leads to Eq. (108) when replaced in Eq. (110).

Derivation of the linear system of equations

Here I show that finding coefficients c_k such that the relation

$$\prod_{j=1}^{N}\hat{x}_j^{n_j} = \sum_{k=1}^{N}c_k\sum_{S\in[N]^k}\left(\sum_{i=1}^{k}\hat{x}_{S_i}^{n_{S_i}}\right)^N \tag{112}$$

holds is equivalent to solving the linear system $A\vec{c}=0$, with $\vec{c}=(c_N, c_{N-1}, \ldots, c_1)$ and A given by

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & \ldots & 0 \\ 1 & 2 & 1 & 0 & \ldots & 0 \\ 1 & 3 & 3 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \binom{N-1}{0} & \binom{N-1}{1} & \binom{N-1}{2} & \binom{N-1}{3} & \cdots & \binom{N-1}{N-1} \end{pmatrix}. \tag{113}$$

Define $Y_j := x_j^{n_j}$ such that Eq. (112) becomes

$$\prod_{j=1}^{N} \hat{Y}_j = \sum_{k=1}^{N} c_k \sum_{S \in [N]^k} \left(\sum_{i=1}^{k} \hat{Y}_{S_i} \right)^N . \tag{114}$$

The expansion of the right-hand side of Eq. (114) produces monomials of the form $\prod_{j=1}^{N} Y_j^{m_j}$, where $\sum_{j=1}^{N} m_j = N$ and the exponents $m_j \geq 0$ are non-negative integers. Each monomial can be uniquely labelled by the vector of exponents $\vec{m} = (m_1, m_2, \ldots, m_N)$. For each polynomial $\left(\sum_{i=1}^{k} \hat{Y}_{S_i} \right)^N$, it follows from the multinomial theorem that the coefficient in front of the monomial $\prod_{j=1}^{N} Y_j^{m_j}$ is always the same, namely the multinomial coefficient $\binom{N}{m_1, m_2, \ldots, m_N}$. For example, the polynomials $(Y_1 + Y_2)^3$ and $(Y_1 + Y_2 + Y_3)^3$ both produce a monomial $Y_1 Y_2^2$ with coefficient $\binom{3}{1,1,0} = 3$. Therefore, the overall coefficient $\chi_{\vec{m}}$ accompanying the monomial $\prod_{j=1}^{N} Y_j^{m_j}$ is given by

$$\chi_{\vec{m}} = \binom{N}{m_1, m_2, \ldots, m_N} \sum_{k=1}^{N} c_k f_k(\vec{m}), \tag{115}$$

where $f_k(\vec{m})$ is the number of times the monomial $\prod_{j=1}^{N} Y_j^{m_j}$ appears in polynomials of k variables. The goal is to find coefficients c_k such that $\chi_{\vec{m}} = 0$ for all $\vec{m}$ except the target case $\vec{m} = (1, 1, \ldots, 1)$. This leads to the equations

$$\sum_{k=1}^{N} c_k f_k(\vec{m}) = 0. \tag{116}$$

Suppose that $\vec{m}$ has ℓ non-zero elements, i.e., the monomial $\prod_{j=1}^{N} Y_j^{m_j}$ contains ℓ variables. The quantity $f_k(\vec{m})$ is then equal to the number of ways in which the remaining $k-\ell$ variables can be selected from the remaining $N-\ell$ ones, which is simply $\binom{N-\ell}{k-\ell}$. Thus, $f_k(\vec{m}) = \binom{N-\ell}{k-\ell}$, which depends only on ℓ, leading to $N-1$ equations for each $\ell = 1, 2, \ldots, N-1$:

$$\sum_{k=1}^{N} c_k \binom{N-\ell}{k-\ell} =: A\vec{c} = 0, \tag{117}$$

with A as in Eq. (113).

References

[1] Seth Lloyd. Almost any quantum logic gate is universal. *Phys. Rev. Lett.*, 75(2):346, 1995.

[2] Seth Lloyd. Universal quantum simulators. *Science*, 273:1073, 1996.

[3] Seckin Sefi and Peter van Loock. How to decompose arbitrary continuous-variable quantum operations. *Phys. Rev. Lett.*, 107:170501, 2011.

[4] Timjan Kalajdzievski and Juan Miguel Arrazola. Exact gate decompositions for photonic quantum computing. *Phys. Rev. A*, 99:022341, 2019.

[5] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997.

[6] Timjan Kalajdzievski, Christian Weedbrook, and Patrick Rebentrost. Continuous-variable gate decomposition for the bose-hubbard model. *Phys. Rev. A*, 97(6):062311, 2018.

[7] Juan Miguel Arrazola, Timjan Kalajdzievski, Christian Weedbrook, and Seth Lloyd. Quantum algorithm for non-homogeneous linear partial differential equations. *Phys. Rev. A*, 100:032306, 2019.

[8] Hoi-Kwan Lau, Raphael Pooser, George Siopsis, and Christian Weedbrook. Quantum machine learning over infinite dimensions. *Phys. Rev. Lett*, 118:080501, 2017.

[9] Seckin Sefi, Vishal Vaibhav, and Peter van Loock. A measurement-induced optical kerr interaction. *Phys. Rev. A*, 88:012303, 2013.

[10] Christopher M. Dawson and Michael A. Nielsen. The solovay-kiteav algorithm. *arXiv:quant-ph/0505030v2*, 2005.

[11] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, 2013.

[12] David P DiVincenzo. Two-bit gates are universal for quantum computation. *Phys. Rev. A*, 51(2):1015, 1995.

[13] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52(5):3457, 1995.

[14] Seth Lloyd and Samuel L. Braunstein. Quantum computation over continuous variables. *Phys. Rev. Lett*, 82:1784, 1999.

[15] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Phys. Rev. A*, 71(2):022316, 2005.

[16] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Asymptotically optimal approximation of single qubit unitaries by clifford and t circuits using a constant number of ancillary qubits. *Phys. Rev. Lett.*, 110(19):190502, 2013.

[17] Vadym Kliuchnikov, Alex Bocharov, and Krysta M Svore. Asymptotically optimal topological quantum compiling. *Phys. Rev. Lett.*, 112(14):140504, 2014.

[18] Vadym Kliuchnikov and Jon Yard. A framework for exact synthesis. *arXiv:1504.04350*, 2015.

[19] Alex Bocharov, Martin Roetteler, and Krysta M Svore. Efficient synthesis of universal repeat-until-success quantum circuits. *Phys. rev. lett.*, 114(8):080502, 2015.

[20] Nicolas C. Menicucci, Peter van Loock, Mile Gu, Christian Weedbrook, Timothy C. Ralph, and Michael A. Nielsen. Universal quantum computation with continuous-variable cluster states. *Phys. Rev. Lett*, 97:110501, 2006.

[21] Mile Gu, Christian Weedbrook, Nicolas C. Menicucci, Timothy C. Ralph, and Peter van Loock. Quantum computing with continuous-variable clusters. *Phys. Rev. A*, 79:062318, 2009.

[22] Christian Weedbrook, Stefano Pirandola, Raul Garcıa Patron, Nicolas J. Cerf, Timothy C. Ralph, Jeffrey H. Shapiro, and Seth Lloyd. Gaussian quantum information. *Rev. Mod. Phys.*, 84:621, 2012.

[23] Naomichi Hatano and Masuo Suzuki. Finding exponential product formulas of higher orders. *Quantum Annealing and Other Optimization Methods (Springer, Berlin)*, page 37, 2005.

[24] Nathan Wiebe, Dominic W. Berry, Peter Hoyer, and Barry C. Sanders. Higher order decompositions of ordered operator exponentials. *J. Phys. A: Math. Theor.*, 43:065203, 2010.

[25] Wilhelm Magnus. On the exponential solution of differential equations for a linear operator. *Communications on pure and applied mathematics*, 7(4):649–673, 1954.

[26] Masuo Suzuki. Generalized trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems. *Commun. math. Phys.*, 51:183, 1976.

[27] Andrew M Childs, Dmitri Maslov, Yunseong Nam, Neil J Ross, and Yuan Su. Toward the first quantum simulation with quantum speedup. *arXiv:1711.10980*, 2017.

[28] Suguru Endo, Qi Zhao, Ying Li, Simon Benjamin, and Xiao Yuan. Mitigating algorithmic errors in hamiltonian simulation. *arXiv:1808.03623*, 2018.

[29] Daniel Gottesman, Alexei Kitaev, and John Preskill. Encoding a qubit in an oscillator. *Phys. Rev. A*, 64:012310, 2001.

[30] Kevin Marshall, Raphael Pooser, George Siopsis, and Christian Weedbrook. Repeat-until-success cubic phase gate for universal continuous-variable quantum computation. *Phys. Rev. A*, 91:032321, 2015.

[31] Petr Marek, Radim Filip, Hisashi Ogawa, Atsushi Sakaguchi, Shuntaro Takeda, Jun ichi Yoshikawa, and Akira Furusawa. General implementation of arbitrary nonlinear quadrature phase gates. *Phys. Rev. A*, 97:022329, 2018.

[32] Koji Azuma, Kiyoshi Tamaki, and Hoi-Kwong Lo. All-photonic quantum repeaters. *Nat. Comm.*, 6:6787, 2015.

[33] Nicolas C. Menicucci. Fault-tolerant measurement-based quantum computing with continuous-variable cluster states. *Phys. Rev. Lett*, 112:120504, 2014.

[34] Raymond Kan. From moments of sum to moments of product. *Journal of Multivariate Analysis*, 99(3):542–554, 2008.

[35] Chris Sparrow, Enrique Martín-López, Nicola Maraviglia, Alex Neville, Christopher Harrold, Jacques Carolan, Yogesh N Joglekar, Toshikazu Hashimoto, Nobuyuki Matsuda, Jeremy L O'Brien, et al. Simulating the vibrational quantum dynamics of molecules using photonics. *Nature*, 557(7707):660, 2018.

[36] Tomasz Sowiński, Omjyoti Dutta, Philipp Hauke, Luca Tagliacozzo, and Maciej Lewenstein. Dipolar molecules in optical lattices. *Phys. Rev. Lett.*, 108:115301, 2012.

[37] Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. Photonic quantum algorithm for monte carlo integration. *arXiv:1809.02579*, 2018.

[38] R.P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467, 1982.

[39] D.S. Abrams and S. Lloyd. Simulation of many-body fermi systems on a universal quantum computer. *Phys. Rev. Lett.*, 79:4, 1997.

[40] Alan Aspuru-Guzik, Anthony D. Dutoi, Peter J. Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309:1704, 2005.

[41] James D. Whitfield, Jacob Biamonte, and Alán Aspuru-Guzik. Simulation of electronic structure hamiltonians using quantum computers. *Molecular Phys.*, 109:735, 2011.

[42] Ryan Babbush, Dominic W. Berry, Yuval R. Sanders, Ian D. Kivlichan, Artur Scherer, Annie Y. Wei, Peter J. Love, and Alan Aspuru-Guzik. Exponentially more precise quantum simulation of fermions in the configuration interaction representation. *Quantum Science and Technology*, 3:015006, 2018.

[43] Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. Exponential algorithmic speedup by quantum walk. *Proc. 35th ACM Symposium on Theory of Computing (STOC 2003)*, page 59, 2003.

[44] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270:359, 2007.

[45] Dominic W. Berry and Andrew M. Childs. Black-box hamiltonian simulation and unitary implementation. *Quantum Information and Computation*, 12:29, 2012.

[46] Andrew M. Childs and Nathan Wiebe. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Information and Computation*, 12:901, 2012.

[47] Stephen P. Jordan, Keith S. M. Lee, and John Preskill. Quantum algorithms for quantum field theories. *Science*, 336:1130, 2012.

[48] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Exponential improvement in precision for simulating sparse hamiltonians. *ACM Symposium on Theory of Computing*, 46:283, 2014.

[49] Guang Hao Low and Isaac L. Chuang. Hamiltonian simulation by uniform spectral amplification. *arXiv:1707.05391*, 2017.

[50] Jeongwan Haah, Matthew B. Hastings, Robin Kothari, and Guang Hao Low. Quantum algorithm for simulating real time evolution of lattice hamiltonians. *arXiv:1801.03922*, 2018.

[51] F. Bemani, R. Roknizadeh, and M. H. Naderi. Quantum simulation of discrete curved space-time by the bose-hubbard model: from analog acoustic black hole to quantum phase transition. *arXiv:1612.09094*, 2017.

[52] Keima Kawaki, Yoshihito Kuno, and Ikuo Ichinose. Phase diagrams of the extended bose-hubbard model in one dimension by monte-carlo simulation with the help of a stochastic-series expansion. *Phys. Rev. B*, 95:195101, 2017.

[53] Yasuyuki Kato and Naoki Kawashima. Quantum monte carlo method for the bose-hubbard model with harmonic confining potential. *Phys. Rev. E*, 79:021104, 2009.

[54] B. Capogrosso-Sansone, N.V. Prokof'ev, and B.V. Svistunov. Phase diagram and thermodynamics of the three-dimensional bose-hubbard model. *Phys. Rev. B*, 75:134302, 2007.

[55] Barbara Capogrosso-Sansone, Sebnem Gunes Soyler, Nikolay Prokof'ev, and Boris Svistunov. Monte carlo study of the two-dimensional bose-hubbard model. *Phys. Rev. A*, 77:015602, 2008.

[56] Masanori Kohno. Mott transition in the two-dimensional hubbard model. *Phys. Rev. Lett.*, 108:076401, 2012.

[57] O. Romero-Isart, K. Eckert, C. Rodo, and A. Sanpera. Transport and entanglement generation in the bose-hubbard model. *J. Phys. A: Math. Theor.*, 40:8019, 2007.

[58] Thierry Giamarchi, Christian Ruegg, and Oleg Tchernyshyov. Bose-einstein condensation in magnetic insulators. *Nat. Phys.*, 4:198, 2008.

[59] Valentin Murg. *Classical and Quantum Simulations of Many-Body Systems.* (doctoral dissertation) Technische Universität München Max-Planck-Institut fur Quantenoptik, 2008.

[60] Werner Krauth. Bethe ansatz for the one-dimensional boson hubbard model. *Phys. Rev. B*, 44:17, 1991.

[61] Yi Kai Liu, Matthias Christandl, and F. Verstraete. Quantum computational complexity of the n-representability problem: Qma complete. *Phys. Rev. Lett.*, 98:110503, 2007.

[62] Tzu Chieh Wei, Michele Mosca, and Ashwin Nayak. Interacting boson problems can be qma hard. *Phys. Rev. Lett.*, 104:040501, 2010.

[63] Andrew M. Childs, David Gosset, and Zak Webb. The bose-hubbard model is qma-complete. *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, page 308, 2014.

[64] Samuel L. Braunstein and Peter van Loock. Quantum information with continuous variables. *Rev. Mod. Phys.*, 77:513, 2005.

[65] Daniel J. Brod and Joshua Combes. Passive cphase gate via cross-kerr nonlinearities. *Phys. Rev. Lett.*, 117:080502, 2016.

[66] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103(15):150502, 2009.

[67] Nathan Wiebe, Daniel Braun, and Seth Lloyd. Quantum algorithm for data fitting. *Physical Review Letters*, 109(5):050505, 2012.

[68] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. Preconditioned quantum linear system algorithm. *Phys. Rev. Lett.*, 110:250504, Jun 2013.

[69] A. Childs, R. Kothari, and R. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017.

[70] Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. Quantum linear system algorithm for dense matrices. *Physical Review Letters*, 120(5):050502, 2018.

[71] Lov K. Grover. A fast quantum mechanical algorithm for database search. *Proceedings, 28th Annual ACM Symposium on the Theory of Computing*, 28:212, 1996.

[72] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *AMS Contemporary Mathematics*, 305:53, 2002.

[73] Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. Quantum computational finance: Monte carlo pricing of financial derivatives. *Phys. Rev. A*, 98:022321, 2018.

[74] M. Szegedy. Quantum speed-up of markov chain based algorithms. *FOCS 04 Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, 45:32, 2004.

[75] Ashley Montanaro. Quantum speedup of monte carlo methods. *Proc. Roy. Soc. Ser. A*, 471:2181, 2015.

[76] G. Xu, A. J. Daley, P. Givi, and R. D. Somma. Turbulent mixing simulation via a quantum algorithm. *AIAA Journal*, 56:687, 2018.

[77] Arun K. Pati, Samuel L. Braunstein, and Seth Lloyd. Quantum searching with continuous variables. *arXiv:quant-ph/0002082*, 2000.

[78] Yariv Yanay and Aashish A. Clerk. Reservoir engineering of bosonic lattices using chiral symmetry and localized dissipation. *Phys. Rev. A*, 98:043615, 2018.

[79] Sarah Mostame, Patrick Rebentrost, Alexander Eisfeld, Andrew J Kerman, Dimitris I Tsomokos, and Alán Aspuru-Guzik. Quantum simulator of an open quantum system using superconducting qubits: exciton transport in photosynthetic complexes. *New Journal of Physics*, 14(10):105013, 2012.