PAPER • OPEN ACCESS

Quantum-circuit design for efficient simulations of many-body quantum dynamics

New Journal of Physics
The open-access journal for physics

Quantum-circuit design for efficient simulations of many-body quantum dynamics

Sadegh Raeisi[1,2,3], **Nathan Wiebe**[1,2] **and Barry C Sanders**[1]

[1] Institute for Quantum Information Science, University of Calgary, AB T2N 1N4, Canada
[2] Institute for Quantum Computing, University of Waterloo, ON N2L 3G1, Canada
E-mail: sadegh.raeisi@gmail.com

New Journal of Physics **14** (2012) 103017 (26pp)
Received 15 May 2012
Published 9 October 2012
Online at http://www.njp.org/
doi:10.1088/1367-2630/14/10/103017

Abstract. We construct an efficient autonomous quantum-circuit design algorithm for creating efficient quantum circuits to simulate Hamiltonian many-body quantum dynamics for arbitrary input states. The resultant quantum circuits have optimal space complexity and employ a sequence of gates that is close to optimal with respect to time complexity. We also devise an algorithm that exploits commutativity to optimize the circuits for parallel execution. As examples, we show how our autonomous algorithm constructs circuits for simulating the dynamics of Kitaev's honeycomb model and the Bardeen–Cooper–Schrieffer model of superconductivity. Furthermore, we provide numerical evidence that the rigorously proven upper bounds for the simulation error here and in previous work may sometimes overestimate the error by orders of magnitude compared to the best achievable performance for some physics-inspired simulations.

[3] Author to whom any correspondence should be addressed.

Contents

1. Introduction

Feynman proposed quantum computing as a means to 'imitate' quantum dynamics in order to overcome apparent intractability of universal quantum simulation on classical computers [1]. He conjectured that a universal quantum simulator (UQS) could efficiently simulate quantum evolution. Lloyd formalized Feynman's concept by employing a Trotter ordered-operator expansion to convert continuous time evolution into a quantum circuit $\mathcal{C}$ comprising unitary quantum gates [2]. The UQS is now also referred to as 'digital quantum simulation', both theoretically [3–6] and experimentally [7]. The adjective 'digital' is used to contrast with the term 'analogue quantum simulation', which aims to emulate evolution of a Hamiltonian $\hat{H}$ in a custom-designed experiment [8–12]. The importance and near-future feasibility of the (digital) UQS, albeit without quantum error correction, drives experimental efforts to create these simulators for restricted types of $\hat{H}$ [7].

We present the first autonomous algorithm to design circuits for simulating the evolution generated by a general n-qubit k-local Hamiltonian $\hat{H}^{(n)}$ within a pre-specified tolerance ϵ. An n-qubit k-local $\hat{H}^{(n)}$ is defined to be a linear combination of m Hamiltonians $\hat{\mathfrak{h}}_j^{(n)}$, each acting on n qubits as an identity operator $\mathbb{1}$ on all but $k \in \text{polylog}(n)$ qubits [13], and polylog(n) is a polynomial function of $\log n$. Our tolerance ϵ is the worst-case two-norm distance between the true evolved state under the specified evolution and the simulated output state, maximized

over all allowed input states. We also show in section 8 that these worst-case error bounds that go into these estimates can overestimate the error for random two-local Hamiltonians by orders of magnitude, which suggests that UQS experiments may be much more feasible than previous simulation work has suggested [14–18].

In our analysis, we consider two independent cases of universal gate sets. One case corresponds to a finite set comprising a single two-qubit entangling gate plus a finite number of one-qubit gates. The second case incorporates a single two-qubit entangling gate plus both discrete and continuously-parameterized single-qubit gates. Strictly speaking only a finite gate set should be permitted for quantum error correction and scalability, but there is a trend in experimental studies to report quantum simulation with continuously-parameterized single-qubit gates. We want our algorithm to be relevant both to the strict case of a finite gate set and to experimental efforts that employ continuous tunability.

Our resultant circuits are not only efficient (meaning that the circuit size scales polynomially with the number of simulated qubits for fixed k) but also uses the smallest number of qubits possible given the size of the system being simulated. Additionally, the circuit size also scales near-optimally with the run-time t of the simulation.

This minimization over space and time costs (number of qubits required in the simulator and number of gates) is important for making quantum simulators as close as possible to practical implementation. Each additional qubit and each additional gate can be challenging to implement in practice so reducing these costs is not only important for proving that the scaling is efficient hence possible in principle but also to reduce the costs to make the simulation feasible with small simulators in the near future. Our minimum run-time algorithm is also improved by parallelizing gates by grouping commuting terms in the Trotter decomposition of the evolutionary operator, thus enhancing the near-term feasibility of the quantum simulator. Our work thus enables feasibility of UQS circuits. Experimental UQS circuits will be valuable to predict resultant states under $\hat{H}$-evolution or to provide UQS-generated states as inputs to quantum algorithms for purposes such as acquiring spectral properties or eigenstates of Hamiltonians [19] or to determine particle scattering, e.g. in a relativistic quantum field theory [20, 21]. The ground state is regarded as the most important eigenstate as it uniquely determines all properties of the system [22] and could be used to solve outstanding condensed-matter problems such as determining the energy gap for general Bardeen–Cooper–Schrieffer (BCS) superconductivity models of *finite* systems [23]. Dynamical simulation algorithms have been devised to simulate ground-state properties via adiabatic state preparation [23] or via dissipative interaction with the environment [4, 5].

2. Algorithm for designing the quantum simulator circuit

2.1. Concept

Our algorithm yields a string representing a quantum circuit that comprises a sequence of quantum gates to simulate k-local Hamiltonian evolution within fixed two-norm distance ϵ. The n-qubit k-local Hamiltonian is expressed in the Pauli operator basis as

$$\hat{H}^{(n)} = \sum_{j=1}^{m} a_j \hat{\mathfrak{h}}_j^{(n)}, \quad \hat{\mathfrak{h}}_j^{(n)} = \otimes_{\ell=1}^{n} \hat{\Xi}_{j\ell}^{(n)}, \tag{1}$$

with $\Xi_{j\ell}^{(n)} \in \{\mathbb{1}, X, Y, Z\}$ such that

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \mathrm{i}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2}$$

The total number of non-identity Pauli operators in each $\hat{\mathfrak{h}}_j^{(n)}$ is at most an n-independent constant k.

Our algorithm is designed to produce a poly(n)-size description of a quantum circuit that implements a unitary $\tilde{U}^{(n)}(t)$ such that

$$\| \exp(-\mathrm{i}\hat{H}^{(n)}t) - \tilde{U}^{(n)}(t) \| \leqslant \epsilon, \tag{3}$$

where $\| \cdot \|$ denotes the two-norm. This condition implies that the trace distance between the ideal evolution and the simulated evolution is at most ϵ for any initial state [17, 18]. Below we discuss the algorithmic input, processing and output.

2.2. Input

The algorithm requires the following inputs:

n: number of qubits in the system;

k: locality parameter of the Hamiltonian;

t: evolution time for the simulation;

ϵ: worst-case two-norm error tolerance (distance) between the true evolved state and the simulated state;

ϖ; specifies which single-qubit gate set to use, namely $\{H, T = Z^{-1/4}\}$ or the continuously parameterized set $\{H, R_z(\theta)\}$, with $Z = R_z(\pi/2)$;

$[\hat{H}^{(n)}]$: bit-string representation of the n-qubit k-local Hamiltonian.

The Hamiltonian $[\hat{H}^{(n)}]$ is entered into the algorithm as a bit string no larger than a poly(n)-size representation for this input to be efficient. Our algorithm accepts the Hamiltonian $\hat{H}^{(n)}$ input as the bit-string representation

$$[\hat{H}^{(n)}] := \left\{ \left(a_j, \boldsymbol{l}_j, \boldsymbol{S}_j\right) ; j = 1, \ldots, m \right\}, \tag{4}$$

with

$$\boldsymbol{l}_j = \left(l_{Xj}, l_{Yj}, l_{Zj}\right) \tag{5}$$

the vector corresponding to the numbers of each type of Pauli operators in $\hat{\mathfrak{h}}_j^{(n)}$ and

$$\boldsymbol{S}_j = \left(\boldsymbol{S}_{Xj}, \boldsymbol{S}_{Yj}, \boldsymbol{S}_{Zj}\right) \tag{6}$$

with $\boldsymbol{S}_{Xj}$, $\boldsymbol{S}_{Yj}$ and $\boldsymbol{S}_{Zj}$ the strings corresponding to the positions of each of the X, Y and Z operators respectively in $\hat{\mathfrak{h}}_j^{(n)}$.

In equation (4), substrings representing Hamiltonians $\hat{\mathfrak{h}}_j^{(n)}$ appear as triplets of strings. These triplets are delimited by parentheses thus ensuring easy parsing of the overall string for $[\hat{H}^{(n)}]$. Whereas a general matrix representation of $\hat{H}^{(n)}$ is exponentially large in n, our representation $[\hat{H}^{(n)}]$ of this k-local Hamiltonian has poly(n) size.

As an example of this encoding, consider the three-qubit Hamiltonian

$$\hat{H}^{(n=3)} = X \otimes X \otimes \mathbb{1} + 2\, Y \otimes Y \otimes \mathbb{1} + 4\, Y \otimes \mathbb{1} \otimes Z \tag{7}$$

with $m = 3$. This Hamiltonian is represented as

$$\begin{aligned} \boldsymbol{a} &= (1, 2, 4), \\ \boldsymbol{l}_1 &= (2, 0, 0), \quad \boldsymbol{S}_{X1} = (1, 2), \\ \boldsymbol{l}_2 &= (0, 2, 0), \quad \boldsymbol{S}_{Y2} = (1, 2), \\ \boldsymbol{l}_3 &= (0, 1, 1), \quad \boldsymbol{S}_{Y3} = (1), \boldsymbol{S}_{Z3} = (3) \end{aligned} \tag{8}$$

and all other strings are empty.

Our algorithm can compute the number of time steps r of equal duration $\frac{t}{r}$ so that the evolution from 0 to t is broken down into a sequence of finite steps. Furthermore our algorithm employs the Trotter–Suzuki (TS) ordered-exponential decomposition [17, 18, 24, 25] of order χ to simulate the evolution sequentially over each of the r time steps. However, our algorithm determines r and χ based on optimizing an upper bound on tolerance, and superior choices of r and χ are possible but not by using our algorithm or any other known algorithm. We provide numerical evidence of this fact, for random two-body Hamiltonians, in section 8.

As the user of our algorithm might wish to employ or test other choices of r and χ, we design the algorithm so that the user can override our choices of r and χ. If our algorithmically determined value of r is overridden, the guarantee that the quantum simulations yields a resultant state within tolerance ϵ no longer holds. Therefore, overriding the algorithm's r comes with the warning that the error in the quantum simulation is unknown. Specifically setting $r = 0$ and $\chi = 0$ causes our algorithm to determine optimal r and χ values based on minimizing the number of gates required to guarantee that the simulated state has two-norm error less than ϵ for any input state. Positive values of r and χ override our algorithmic determination of these parameters and use the input values instead.

The purpose of the input variable ϖ is to enable circuit design that strictly uses a finite gate set in accordance with principles of quantum error correction and scalability [26] or to include a continuously-parameterized single-qubit gate, namely rotations by θ around the z-axis using unitary operator $R_z(\theta)$, in accordance with prevalent experimental practice [7]. More generally one could consider the case that ϖ is any desired universal gate set, but here we restrict our attention to just two single-qubit gates, either $\{H, T = Z^{-1/4}\}$ or $\{H, R_z(\theta)\}$, so ϖ is a binary, or logical, variable. In our algorithm, both of these sets are accompanied by two-qubit controlled-not gate CNOT_{ij} with the i labeling the control qubit and j labeling the target qubit.

2.3. Output

Our algorithm yields an output string $[\mathcal{C}]$ that represents the resultant quantum circuit. This quantum circuit is a sequence of quantum gates that simulate the dynamics of the quantum system for an arbitrary input state. In our algorithm the basic quantum gates are members either of the finite-size universal instruction set $\{H, T, \mathrm{CNOT}\}$ or the continuously-parameterized set $\{H, R_z(\theta), \mathrm{CNOT}\}$.

The single-qubit gates are represented in the output as a string of the form 'Hx' or 'Tx' with x a bit string labeling the qubit acted upon by gate T or H, respectively. These gates, along with the identity $\mathbb{1}$, which is not explicitly stated in the circuit, can be parallelized and concatenated to form circuits. For example, the string $H1T2$ represents a circuit that first

performs the Hadamard operation on qubit 1 then a $\pi/8$–gate on qubit 2. The CNOT operation is represented similarly with the string 'CNOTx,y' representing the quantum operation CNOT$_{xy}$. For the case of the gate set that includes continuously parameterized single-qubit z-rotation gates, $R_z(\theta)$, in the output, The action of $R_z(\theta)$ on qubit x is represented as the string '$RZ[\theta]$, x' where $[\theta]$ is a string representing the rotation angle θ.

2.4. Processing

Our circuit-design algorithm proceeds through three stages.

1. *Hamiltonian sorting algorithm:* Mutually commuting terms in $H^{(n)}$ are grouped together resulting in parallelized quantum simulation that reduces total runtime.
2. *TS algorithm:* $U^{(n)}(t) := \exp\{-\mathrm{i}\hat{H}^{(n)}t\}$ is decomposed into a sequence of exponentials of Pauli operators.
3. *Circuit-design algorithm for Pauli-exponentials:* Determine the quantum circuit and convert into output string $[C]$.

These algorithmic stages are described in the following sections.

2.5. Summary

An algorithm consists of input and output with the output obtained by processing the input. We have been careful to discuss the input and output as bit strings as our algorithm is classical and runs on a classical computer. The output of the algorithm is a design procedure to construct a quantum simulator circuit that would simulate the evolution of a quantum state. In the following sections we describe the algorithmic stages.

3. Trotter–Suzuki (TS) formulae

In this section we discuss the second stage of the algorithm, which concerns decomposing the TS ordered-operator exponential into a sequence of exponentials of tensor products of Pauli operators. The second stage of the algorithm is discussed first because the first stage groups these TS terms so understanding the second stage helps to understand the terms being grouped in the first stage.

We use the TS method to determine a product of exponentials of $\hat{\mathfrak{h}}_j^{(n)}$ that approximates $U^{(n)}(t)$ within two-norm distance ϵ [17, 18, 24]. The total time of evolution t is divided into r time intervals each of equal duration

$$\Delta t = \frac{t}{r}. \tag{9}$$

For $\tilde{U}_\chi^{(n)}$ the χth-order TS iterate approximating the n-qubit unitary evolution $U^{(n)}$, the distance between the 'true' evolution operator $U^{(n)}(t)$ and the TS approximated evolution operator is

$$\left\| U^{(n)}(t) - U_\chi^{(n)}\,(\Delta t)^r \right\| \in O\left(\frac{t^{2\chi+1}}{r^{2\chi}}\right). \tag{10}$$

The iterative TS formula for generating U_χ is well known [24]. The formulae are widely used in quantum simulation algorithms because they generate an approximation

$$U_\chi^{(n)}(\Delta t) = \exp\left\{-\mathrm{i}a_{j_1}\hat{\mathfrak{h}}_{j_1}^{(n)}t_1\right\}\exp\left\{-\mathrm{i}a_{j_2}\hat{\mathfrak{h}}_{j_2}^{(n)}t_2\right\}\cdots\exp\left\{-\mathrm{i}a_{j_M}\hat{\mathfrak{h}}_{j_M}^{(n)}t_M\right\}, \tag{11}$$

which is a product of unitary evolutions, for Hamiltonians $\hat{\mathfrak{h}}_{j_1}^{(n)}$ represented by the sequence j_i and a sequence of times t_i. The TS formulae comprises a sequence of exponentials of Pauli operators that have simple circuits for quantum computer implementation [26].

Specifically, the approximation $U_\chi^{(n)}(\frac{t}{r})$ is constructed iteratively for the Hamiltonian $H^{(n)} = \sum_{j=1}^m a_j\hat{\mathfrak{h}}_j^{(n)}$ via

$$\begin{aligned} U_1^{(n)}(\Delta t) &= \prod_{j=1}^{m}\exp\left\{-\mathrm{i}a_j\hat{\mathfrak{h}}_j^{(n)}\frac{\Delta t}{2}\right\}\prod_{j=m}^{1}\exp\left\{-\mathrm{i}a_j\hat{\mathfrak{h}}_j^{(n)}\frac{\Delta t}{2}\right\}, \\ U_p^{(n)}(\Delta t) &= \left[U_{p-1}^{(n)}\left(\frac{s_p t}{r}\right)\right]^2 U_{p-1}^{(n)}\left((1-4s_p)\frac{t}{r}\right)\left[U_{p-1}^{(n)}\left(\frac{s_p t}{r}\right)\right]^2 \end{aligned} \tag{12}$$

with

$$s_p = \frac{1}{4-4^{1/(2p-1)}} \tag{13}$$

and the integer p obeys $1 < p \leqslant \chi$. We emphasize this form of the TS formula because it is important for our grouping algorithm that the order of the exponentials in the product formula matches the order of the exponentials in the Hamiltonian.

We express this TS stage of the algorithm as an outline of a computer program. The program's input is the bit-string representation $[\hat{H}^{(n)}]$ of the n-qubit Hamiltonian, the desired order of the Suzuki iteration χ, the evolution time t and the number of intervals r, which together yield the time step Δt (9). The program's output bit-string representation for the χth-order approximation to the true evolution:

$$[U_\chi^{(n)}(\Delta t)] \equiv (a_{j_M}, \boldsymbol{l}_{j_M}, \boldsymbol{S}_{j_M}, t_M)(a_{j_{M-1}}, \boldsymbol{l}_{j_{M-1}}, \boldsymbol{S}_{j_{M-1}}, t_{M-1})\cdots(a_{j_1}, \boldsymbol{l}_{j_1}, \boldsymbol{S}_{j_1}, t_1) \tag{14}$$

with $M = 2m5^{\chi-1}$ following from the recursive form of the TS formulae (12) [24]. The representation in (14) stores each exponential in the approximation as a sequence of strings that represent a Hamiltonian term $a_j\mathfrak{h}_j^{(n)}$ and then the duration of the evolution step (stored to finite precision). Our representation stores the exponentials in order of their execution; although, in this case, the symmetry of the TS formulae implies that we could also store them in reverse order without changing the result. The TS algorithm that generates a TS approximation of the form (14) is given by algorithm 1.

The performance of the resulting simulation depends strongly on the chosen values for r and χ. If $r = 0$ or $\chi = 0$, our program determines suitable values of r or χ; otherwise this program uses the user-supplied values. Given a specified value of χ, then

$$r = \left\lceil \frac{(2m(5/3)^{\chi-1}\chi(\max_j|a_j|t))^{1+1/2\chi}}{(\epsilon/2)^{1/2\chi}} \right\rceil \tag{15}$$

is optimal [14, 16], which guarantees that [14]

$$\left\| U^{(n)}(t) - \left(\tilde{U}^{(n)}(\Delta t)\right)^r \right\|_2 \leqslant \frac{\epsilon}{2} \tag{16}$$

Algorithm 1. TS algorithm.

Input:

$[\hat{H}^{(n)}]$: bit-string representation of the Hamiltonian

Δt: time duration

χ: iteration order of Suzuki's method [24]

Output:

SuzInt: array of exponentials.

function TROTTERSUZUKI($[\hat{H}^{(n)}]$, Δt, χ)

return SuzInt $\leftarrow$ sequence of exponentials $[U_{\chi}^{(n)}(\Delta t)]$ (14) using Suzuki's procedure [24].

end function

provided that

$$\epsilon \leqslant 2m\chi(5/3)^{\chi-1} \max_j |a_j| t. \tag{17}$$

We employ the value of r in (15) as the default value of r for the algorithm and take our default value of χ to be

$$\chi = \left\lceil \sqrt{\frac{\log_{25/3}(m \max_i |a_i| t/\epsilon)}{2}} \right\rceil \tag{18}$$

because it causes the number of operations in the simulation to scale nearly linearly with t [14].

The value of r given in (15) can be larger than necessary for certain Hamiltonians (as shown in section 8). If a guarantee that the error is less than the tolerance ϵ is not required, then choosing r smaller than the optimal value given in (15) could suffice and thereby reduce runtime.

4. Hamiltonian sorting algorithm

Now we return to the first stage of the algorithm, which aims to reduce runtime by grouping TS terms based on generation by commuting Hamiltonians. In other words, TS terms are grouped together to parallelize the quantum simulation circuit. The benefits of grouping terms and exploiting parallelism is most notable in the case of physically local Hamiltonians, where we show that parallelism typically leads to a near-quadratic improvement to the scaling of the time required to simulate the quantum system's evolution.

The algorithm achieves this grouping by decomposing the Hamiltonian (1) into $\bar{m}$ groups of terms as

$$\hat{H}^{(n)} = \sum_{j=1}^{\bar{m}} \hat{\mathfrak{g}}_j^{(n)}, \quad \hat{\mathfrak{g}}_j^{(n)} = \sum_{l \in G_j} a_l \hat{\mathfrak{h}}_l^{(n)}, \tag{19}$$

such that all $\hat{\mathfrak{h}}_l^{(n)} \in G_j$ mutually commute.

Algorithm 2. Hamiltonian sorting algorithm.

Input:

$[\hat{H}^{(n)}]$: bit-string representation of the Hamiltonian

n: number of qubits

m: number of Hamiltonian terms summed to make $[\hat{H}^{(n)}]$

Output:

$[\hat{H}^{(n)}_{\text{sorted}}]$: sorted representation of the Hamiltonian with mutually commuting terms combined in groups

function SORTH($[\hat{H}^{(n)}]$, n, m)
  $m_s \leftarrow 1$.
  $G_1 \leftarrow (a_1, \boldsymbol{l}_1, \boldsymbol{S}_1)$.
  for j From 2 to m **do**
    isAssigned $\leftarrow 0$.
    for p from 1 to m_s **do**
      if isAssigned $= 0$ **then** ▷ Checks if term commutes with terms in G_p
        isAssigned $\leftarrow 1$ if $\sum_{v \neq w} \left| \boldsymbol{S}_{vi} \cap \boldsymbol{S}_{wj} \right|$ for each $(a_i, \boldsymbol{l}_i, \boldsymbol{S}_i)$ in G_p.
        if isAssigned $= 1$ **then**
          $G_p \leftarrow$ Concatenation of G_p with $(a_j, \boldsymbol{l}_j, \boldsymbol{S}_j)$. ▷ Assigns term to G_p.
        end if
      end if
    end for
    if isAssigned $= 0$ **then** ▷ Checks if new 'group' of commuting Hamiltonians is needed
      $m_s \leftarrow m_s + 1$.
      $G_{m_s} \leftarrow (a_j, \boldsymbol{l}_j, \boldsymbol{S}_j)$.
    end if
  end for
  return $[H_{\text{sorted}}] \leftarrow$ Concatenation of $G_1, \ldots, G_{m_s}$.
end function

The TS algorithm can then be used to determine a sequence of exponentials of $\hat{\mathfrak{g}}^{(n)}_j$, namely the sum of the terms in each commuting set that simulates

$$\exp\left(-\mathrm{i}\hat{H}^{(n)}t\right) = \exp\left(-\mathrm{i}\sum_{j=1}^{\bar{m}} \hat{g}^{(n)}_j t\right) \tag{20}$$

within error ϵ. Product-formula approximations are not needed to decompose the exponential of each group into a product of exponentials of Pauli-operations as each $\hat{\mathfrak{h}}^{(n)}_j$ in any given group mutually commutes. In other words

$$\exp\left(-\mathrm{i}\hat{\mathfrak{g}}^{(n)}_i t\right) = \exp\left(-\mathrm{i}\sum_{j\in G_i} \hat{\mathfrak{h}}^{(n)}_j t\right) = \prod_{j\in G_i} \exp\left(-\mathrm{i}\hat{\mathfrak{h}}^{(n)}_j t\right) \tag{21}$$

for any value of t.

Finally, we note that explicitly grouping the terms in $\hat{H}^{(n)}$ into $\hat{\mathfrak{g}}^{(n)}$ is unnecessary. Instead it suffices to sort the terms in the Hamiltonian by group membership (i.e. terms that are assigned to group G_1 appear first in $[\hat{H}^{(n)}]$, then terms in G_2 appear next and so forth). If we use (12) to construct the TS formulae, then the resulting simulation will sort the steps in the simulation into commuting groups of operations. As commuting operations can often be executed in parallel, this procedure reduces the time required to execute the circuit in systems that can use parallelism.

In the first stage of the algorithm our program accepts the string $\boldsymbol{S}_i$, as introduced in section 2, to determine if two terms in $\hat{H}^{(n)}$ commute. The Hamiltonians $\hat{\mathfrak{h}}_i^{(n)}$ and $\hat{\mathfrak{h}}_j^{(n)}$ commute if and only if, for the dummy variables $v, w \in \{X, Y, Z\}$,

$$\sum_{v \neq w} \left| \boldsymbol{S}_{vi} \cap \boldsymbol{S}_{wj} \right| \equiv 0 \ (\mathrm{mod}\ 2), \tag{22}$$

where $|\bullet|$ denotes the size of a set $\bullet$.

As a clarifying example, consider the Hamiltonian (7). The first two terms in (7) commute because of the anti-commutativity of Pauli-operators and because both terms have differing actions on an even number of qubits. Criterion (22) also tells us that they commute because

$$\sum_{v \neq w} |\boldsymbol{S}_{v1} \cap \boldsymbol{S}_{w2}| = |\boldsymbol{S}_{x1} \cap \boldsymbol{S}_{y2}| = 2 \equiv 0 \ (\mathrm{mod}\ 2). \tag{23}$$

On the other hand, the second and third terms do not commute as

$$\sum_{v \neq w} |\boldsymbol{S}_{v2} \cap \boldsymbol{S}_{w3}| = |\boldsymbol{S}_{y2} \cap \boldsymbol{S}_{z3}| = 1 \equiv 1 \ (\mathrm{mod}\ 2). \tag{24}$$

Criterion (22) accounts for the commutativity of Hamiltonians that act on disjoint sets of qubits as well as other commuting Hamiltonians such as the star and plaquette operators in the toric code [27]. A proof of the general validity of (22) as a criterion for commutativity is given in appendix. If desired, a more restrictive grouping condition

$$\sum_{v \neq w} |\boldsymbol{S}_{vi} \cap \boldsymbol{S}_{wj}| \equiv 0 \tag{25}$$

can be used only to group together operations that have actions on disjoint sets of qubits.

The criterion for commutativity in (22) can be used to find an efficient classical algorithm for grouping $\{\hat{\mathfrak{h}}_j^{(n)}\}$ into groups of mutually commuting Hamiltonians, which we describe below formally. This grouping algorithm is efficient because m is polynomially large in n for local-Hamiltonians and (22) can be efficiently evaluated.

The depth of the resulting quantum circuit depends on the value of $\bar{m}$ for the Hamiltonian. The reductions in the depth vary with the number of groups required for the Hamiltonian in question. The question can, however, be addressed for the cases of generic k-local or physically k-local Hamiltonians (by generic we mean k-local Hamiltonians that include every possible p-body interaction for $p \leqslant k$.).

We estimate the worst-case scaling of $\bar{m}$ for generic k-local $\hat{H}^{(n)}$ by using the more restrictive grouping condition that $\hat{\mathfrak{h}}_i^{(n)}$ and $\hat{\mathfrak{h}}_j^{(n)}$ are assigned to the same group only if condition (25) is satisfied. This requirement will generically result in a smaller value of $\bar{m}$ than our grouping algorithm will yield. In this case, at most $\lfloor n/k \rfloor$ k-body terms can be assigned to each group for k-local interactions. Terms with $(k-1)$-body interactions (or fewer)

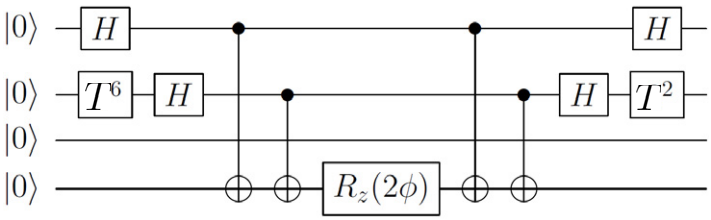

Figure 1. Quantum circuit for implementing $\exp(-\mathrm{i}\phi X \otimes Y \otimes \mathbb{1} \otimes Z)$ for an arbitrary dimensionless evolution time ϕ with H the Hadamard gate, T^{ℓ} a concatenation of ℓ $\pi/8$ gates and $R_Z(2\phi)$ a qubit rotation of ϕ about Z.

can be neglected because n is assumed to be large and the vast majority of terms in the k-local Hamiltonian are k-body. Specifically, there are $O(n^{k-1})$ terms with only $(k-1)$-body interactions or fewer and $O(n^k)$ terms with k-body interactions, which means that we can neglect terms that are only $(k-1)$-local for generic Hamiltonians in the limit of large n.

The number of k-body Hamiltonians that can be assigned to each group before the Hamiltonians violate the grouping criterion (22), scales as $\Theta(n/k)$, with Θ the Bachmann–Landau notation for indicating that a function of this order is asymptotically bounded above and below by n/k up to multiplicative constants. For k a constant

$$\bar{m} \in O\left(\frac{n^k}{n/k}\right) = O(n^{k-1}). \tag{26}$$

Physically local Hamiltonians are constrained such that each qubit interacts with at most a constant number of qubits. This implies that there are $O(n)$ k-body terms present in physically k-local Hamiltonians. Therefore, the number of groups required for physically-local Hamiltonians scales as

$$\bar{m} \in O\left(\frac{n}{n/k}\right) = O(1), \tag{27}$$

which we will see constitutes a nearly-quadratic reduction in the circuit depth for some simulations.

5. Implementing TS formulae

The main primitive element of the circuit-design algorithm is a basic circuit element $\mathcal{C}_\ell$ corresponding to a particular sequence of gates in our gate set to simulate evolution due to each $\exp\{-\mathrm{i}a_{j_\ell}\hat{\mathfrak{h}}^{(n)}_{j_\ell}t_\ell\}$ in the output of the subroutine described in section 3. The circuit construction that we use is an optimized version of that presented in [26]. Underpinning this primitive $\mathcal{C}_\ell$ is the conversion of Pauli gate operations to operations in our gate set [4–6].

The simplest way to explain this step is by example:

$$U_\ell^{(n=4)}(\phi) = \exp\left(-\mathrm{i}\phi X \otimes Y \otimes \mathbb{1} \otimes Z\right) \tag{28}$$

shown in figure 1. This circuit allows for a continuously parameterized phase-rotation gate $R_z(2\phi) = \exp(-\mathrm{i}\phi Z)$, but of course a proper quantum algorithm would work with a finite gate set. However, the gate $R_Z(2\phi)$ can be reduced to a finite gate set using the constructive version of the Solovay–Kitaev algorithm [28]. Our algorithm takes a logical variable, $\varpi \in \{0, 1\}$ as an

input that specifies the gate set from which the gates in the output should be drawn and uses the Dawson–Nielsen algorithm [28] to convert the continuous rotation gates into discrete gates if gate set 0 is chosen. We label the gate sets

$$\varpi = 0\text{: } \{H, T, \text{CNOT}\},$$
$$\varpi = 1\text{: } \{H, R_z(\theta), \text{CNOT} : \theta \in [0, 2\pi)\}.$$

The circuit primitive for $\mathcal{C}_\ell$, as illustrated in figure 1, is constructed by first choosing one of the system qubits to be the 'parity qubit'. The role of this qubit is to track the parity of qubits affected by $\hat{\mathfrak{h}}_{j_\ell}^{(n)}$ when expressed in the eigenbasis of $\hat{\mathfrak{h}}_{j_\ell}^{(n)}$. This parity is required to perform $U_\ell^{(n)}(a_{j_\ell} t_\ell)$ using diagonalization [4–6]. The parity qubit is always chosen to be the qubit with the largest label amongst this set that is non-trivially affected by the Hamiltonian. We say that a qubit is trivially affected by a Hamiltonian if it acts as the identity on that qubit. As an example, the fourth (bottom) qubit is chosen to be the parity qubit for the Hamiltonian in figure 1.

The method we use to construct the simulation circuit is given in algorithm 3. Specifically this stage of the algorithm produces a bit-string representation for a circuit approximating $\exp\{-\mathrm{i} a_i \mathfrak{h}_i^{(n)} t_i\}$ where $a_i \mathfrak{h}_i^{(n)}$ is a term in $\hat{H}^{(n)}$. The algorithm for this is provided below.

6. Main algorithm

6.1. Complete procedure

We now construct the circuit-design algorithm, which employs the programs described in the previous three sections. The algorithm begins by using algorithm 2 to sort the terms in the Hamiltonian, which ensures that neighboring entries in the list of terms that comprise the Hamiltonian commute (if possible). The next step uses the TS algorithm to find a sequence of simulations of the $\hat{\mathfrak{h}}_{j_\ell}^{(n)}$ that approximates $\mathrm{e}^{-\mathrm{i}\hat{H}^{(n)}t}$. The final step utilizes algorithm 3 to find a quantum circuit that approximates each of the $\exp\{-\mathrm{i}\hat{\mathfrak{h}}_{j_\ell}^{(n)} t_\ell\}$ to yield a complete description of the overall simulation circuit. The procedure is described in greater detail in algorithm 4.

The efficiency of our main algorithm depends on whether r and $2m5^{\chi_0-1}$ are polynomially large. It is straightforward to see by substitution that the default values used for both of these quantities scale polynomially with the simulation parameters, and hence is efficient. On the other hand, if the default values of r and χ are not used then the above algorithm may not be efficient.

6.2. Cost estimates

We now provide upper bounds for the scaling of the circuit-size of the circuits yielded by our design algorithm using the default values χ_0 and r_0, which are chosen to guarantee that the simulation time scales near-linearly with t and the error is at most $\epsilon/2$ respectively. There are three costs that we consider: the number of gates in the resulting circuit, N_{op}, the time required to execute the circuit using parallelism, τ, and the number of qubits required which in our case is trivially n. We assess the remaining costs N_{op} and τ below.

The value of N_{op} is bounded above by the number of circuit primitives, $\mathcal{C}_\ell$, needed to simulate the evolution multiplied by the maximum cost of implementing a circuit primitive.

Algorithm 3. Circuit-design algorithm for Pauli-exponentials.

Input:

$(a_i, \boldsymbol{l}_i, \boldsymbol{S}_i, t_i)$: bit-string representation of the exponential
ϖ: determines which gate set should be used
δ: error-tolerance for the Solovay–Kitaev algorithm

Output:

$[\mathcal{C}_i]$: simulation circuit for $\exp(-\mathrm{i}\hat{\mathfrak{h}}_i^{(n)} t_i)$

function PCircuit($(\boldsymbol{a}_i, \boldsymbol{l}_i, \boldsymbol{S}_i, t_i), \varpi, \delta$)

$[\mathcal{C}_i] \leftarrow \emptyset$. ▷ Sets $[\mathcal{C}_i]$ to the empty-string.

for each string ℓ in $\boldsymbol{S}_{i,x}$ **do**

$[\mathcal{C}_i] \leftarrow [\mathcal{C}_i] H \ell$. ▷ Concatenates $[\mathcal{C}_i]$ with Hadamard on qubits that $\hat{\mathfrak{h}}_i^{(n)}$ acts as X on.

end for

for each string ℓ in $\boldsymbol{S}_{i,y}$ **do**

$[\mathcal{C}_i] \leftarrow [\mathcal{C}_i] T \ell T \ell T \ell T \ell T \ell T \ell H \ell$. ▷ Applies diagonalizing rotation to each qubit on which the term acts as Y.

end for

$\ell_{\max} \leftarrow \max(\ell \in \boldsymbol{S}_i)$.

for each ℓ in $\boldsymbol{S}_i \setminus \{\ell_{\max}\}$ **do** ▷ Identifies parity qubit.

$[\mathcal{C}_i] \leftarrow [\mathcal{C}_i]\mathrm{CNOT}\ell, \ell_{\max}$.

end for

if ϖ=1 **then**

SK$\leftarrow$ output of the Solovay–Kitaev algorithm for $R_z(2a_i t_i)$ acting on qubit $\ell_{\max}$ with error tolerance δ.

$[\mathcal{C}_i] \leftarrow [\mathcal{C}_i]\mathrm{SK}$.

else

$[\mathcal{C}_i] \leftarrow [\mathcal{C}_i]\mathrm{RZ}(2a_i t_i), \ell_{\max}$.

end if

for each ℓ in $\boldsymbol{S}_i \setminus \{\ell_{\max}\}$ **do**

$[\mathcal{C}_i] \leftarrow [\mathcal{C}_i]\mathrm{CNOT}\ell, \ell_{\max}$.

end for

for each string ℓ in $\boldsymbol{S}_{i,y}$ **do**

$[\mathcal{C}_i] \leftarrow [\mathcal{C}_i] H \ell T \ell T \ell$.

end for

for each string ℓ in $\boldsymbol{S}_{i,x}$ **do**

$[\mathcal{C}_i] \leftarrow [\mathcal{C}_i] H \ell$.

end for

return $[\mathcal{C}_i]$.

end function

The cost for implementing each circuit primitive $\mathcal{C}_\ell$ for a k-local $\hat{H}$ requires at most $10k$ single-qubit gates, $2k-2$ CNOT gates and one R_Z rotation. This worst-case estimate is given by the cost of simulating a $\hat{\mathfrak{h}}_j^{(n)}$ that acts as Y on k qubits because Y-interactions are the most expensive to simulate using our algorithm for finding $\mathcal{C}_\ell$. The Solovay–Kitaev algorithm of Dawson and Nielsen [28] gives the cost of implementing the continuous gate $R_Z(2\phi)$ within

Algorithm 4. Main algorithm.

Input:

$[\hat{H}^{(n)}]$: bit-string representation of the Hamiltonian
n: the number of qubits
t: evolution time
ϵ: error tolerance
r: number of time steps ▷ $r = 0$ guarantees error is at most ϵ
χ: iteration order of TS formula ▷ $\chi = 0$ guarantees near-linear time scaling.
ϖ: logical value that indicates whether the discrete or continuous gate set is used

Output:

$[\mathcal{C}]$: simulation circuit for $\exp(-\mathrm{i}\hat{H}_i^{(n)}t)$

function MAIN($[\hat{H}^{(n)}], n, t, r, \chi, \epsilon, \varpi$)
  $[\mathcal{C}_{\text{temp}}] \leftarrow \emptyset$.
  $m \leftarrow$ number of terms in $\hat{H}^{(n)}$.
  $[\hat{H}^{(n)}] \leftarrow \text{SortH}([\hat{H}^{(n)}], n, m)$. ▷ See algorithm 2.
  $a_{\max} \leftarrow \max_i |a_i|$.
  if $\chi = 0$ **then**
    $\chi \leftarrow \left\lceil \sqrt{\frac{\log_{25/3}\left(\frac{ma_{\max}t}{\epsilon}\right)}{2}} \right\rceil$. ▷ Computes default value of χ
  end if
  if $r = 0$ **then**
    $r \leftarrow \left\lceil \frac{2(2m(5/3)^{\chi-1}\chi a_{\max}t)^{1+1/(2\chi)}}{(\epsilon/2)^{1/(2\chi)}} \right\rceil$. ▷ Computes default value of r
  end if
  if $\epsilon > 2m\chi(5/3)^{\chi-1}\max_j |a_j|t$ **then**
    $\epsilon \leftarrow 2m\chi(5/3)^{\chi-1}\max_j |a_j|t$.
  end if
  SuzInt $\leftarrow$ TrotterSuzuki($[\hat{H}^{(n)}], \frac{t}{r}, \chi$). ▷ See algorithm 1.
  for $j = 1$ to $2m5^{\chi-1}$ **do** ▷ Finds circuit for one time step.
    $[\mathcal{C}_{\text{temp}}] \leftarrow [\mathcal{C}_{\text{temp}}]\text{PCircuit}\left(\text{SuzInt}(j), \varpi, \frac{\epsilon}{(4m5^{\chi-1}r)}\right)$. ▷ See algorithm 3.
  end for
  $[\mathcal{C}] \leftarrow [\mathcal{C}_{\text{temp}}]$.
  for $j = 1$ to $r - 1$ **do** ▷ Finds complete simulation circuit.
    $[\mathcal{C}] \leftarrow [\mathcal{C}][\mathcal{C}_{\text{temp}}]$.
  end for
  return $[\mathcal{C}]$.
end function

tolerance $\epsilon/(4m5^{\chi_0-1}r_0)$ as

$$N_{\text{SK}} \in O\left(\log^4\left(\frac{4m5^{\chi_0-1}(2m(5/3)^{\chi_0-1}\chi a_{\max}t)^{1+1/(2\chi_0)}}{2\epsilon(\epsilon/2)^{1/(2\chi_0)}}\right)\right). \tag{29}$$

As

$$\chi_0 = \left\lceil \sqrt{\frac{\log_{25/3}(ma_{\max}t/\epsilon)}{2}} \right\rceil, \tag{30}$$

we have that $\log(5^{\chi_0}) \in O(\sqrt{\log(ma_{\max}t/\epsilon)})$. Then using the properties of logarithms, we find that

$$N_{\mathrm{SK}} \in O\left(\log^4\left(\frac{m\max_i|a_i|t}{\epsilon}\right)\right). \tag{31}$$

The total number of operations used to implement each primitive circuit $\mathcal{C}_\ell$ within tolerance ϵ is thus $O(k+N_{\mathrm{SK}})$. As $\mathcal{C}=\mathcal{C}_M\cdots\mathcal{C}_1$, the total number of operations in $\mathcal{C}$ scales as $O(M(k+N_{\mathrm{SK}}))$. Finally, $U^{(n)}(t)$ is simulated by $\mathcal{C}^r$ so the total number of operations scales as

$$N_{\mathrm{op}} \in O\left(2m5^{\chi_0-1}r_0(k+N_{\mathrm{SK}})\right). \tag{32}$$

We then substitute the value of r_0 into this expression to find that,

$$N_{\mathrm{op}} \in \frac{(k+N_{\mathrm{SK}})m^{2+o(1)}\,(\max_i|a_i|t)^{1+o(1)}}{\epsilon^{o(1)}}, \tag{33}$$

with $\epsilon^{o(1)}$, $m^{o(1)}$ and $(\max_i|a_i|t)^{o(1)}$ representing quantities that scale sub-polynomially but not quite poly-logarithmically. As N_{SK} varies sub-polynomially with respect to all parameters and k is a constant, $N_{\mathrm{SK}}+k$ can be incorporated into $(m\max_i|a_i|t/\epsilon)^{o(1)}$ in relation (33). This leads to the conclusion that

$$N_{\mathrm{op}} \in \frac{(k+N_{\mathrm{SK}})m^{2+o(1)}\,(\max_i|a_i|t)^{1+o(1)}}{\epsilon^{o(1)}} \in \frac{m^{2+o(1)}\,(\max_i|a_i|t)^{1+o(1)}}{\epsilon^{o(1)}}. \tag{34}$$

The performance of our circuit-design algorithm is enhanced if a reasonable extra restriction is placed on the k-local $\hat{H}$. The upper bound m is different for k-local versus physically k-local $\hat{H}$ as we now see. For k-local $\hat{H}^{(n)}$,

$$m \leqslant \sum_{q=1}^{k} 3^q \binom{n}{q} \in O(n^k), \tag{35}$$

because there are at most $3^q\binom{n}{q}$ q-body terms in $\hat{H}^{(n)}$ for $q=1,\ldots,k$. The scaling $m\in O(n^k)$ arises from standard inequalities for binomial sums and, as k is a constant, then so is 3^k. If $\hat{H}^{(n)}$ is physically k-local, $m\in O(n)$ because each qubit interacts with at most a constant number of neighbors.

We can then eliminate m from (33) by noting that, if $\hat{H}^{(n)}$ is k-local, then $m\in O(n^k)$ for k a constant, and

$$N_{\mathrm{op}} \in \frac{n^{k(2+o(1))}\,(\max_i|a_i|t)^{1+o(1)}}{\epsilon^{o(1)}}. \tag{36}$$

If $\hat{H}^{(n)}$ is physically k-local and k is constant, then $m\in O(n)$. We substitute this scaling into (33) to obtain

$$N_{\mathrm{op}} \in \frac{n^{2+o(1)}\,(\max_i|a_i|t)^{1+o(1)}}{\epsilon^{o(1)}}. \tag{37}$$

Comparing (36) to (37) shows that the simulation cost is dramatically reduced for $\hat{H}^{(n)}$ physically k-local rather than just k-local. This cost reduction does not occur from a modification of the algorithm, but rather a more careful costing of the performance of our algorithm.

In fact the circuits generated by our algorithm are optimal, or near-optimal, in three distinct ways. Firstly, they exhibit near-optimal scaling with t because linear scaling is known to be a

lower bound for general quantum simulation [17, 18, 29]. Secondly, they have optimal space complexity because a minimum of n-qubits of memory is needed to simulate the quantum dynamics of an n qubit system. Finally, the n-scaling of (36) and (37) is unlikely to be surpassed by other general purpose TS-based simulation algorithms because the scaling with n is derived from the value of m for the Hamiltonian [17, 18].

The fact that better scaling with n cannot be obtained by using a superior decomposition method for the Hamiltonian follows from Vizing's edge-coloring graph algorithm [30], which states that a graph with maximum degree d cannot be colored using fewer than d colors. This implies that a d-sparse Hamiltonian can be decomposed into at best d one-sparse matrices. A k-local Hamiltonian can be at most $O(n^k)$ sparse, which implies that $O(n^k)$ terms will be present in the Hamiltonian using the optimal decomposition method. This value of m coincides with the value that our algorithm finds for simulating k-local Hamiltonians; hence our algorithm is unlikely to be significantly surpassed by other algorithms that use similar strategies.

Now we will examine the scaling of the time required to implement the resultant quantum circuits on quantum computers that can exploit parallelism. Without grouping, the depth of the quantum circuits yielded by our algorithm scales with m is at worst $m^{2+o(1)}$. Although our grouping step does not change the circuit size, it causes the depth of the resulting circuits to scale as $\bar{m}m^{1+o(1)}$, where $\bar{m}$ may be smaller than m.

The factor of $m^{1+o(1)}$ remaining in the scaling comes from the upper-bound used to estimate error in the TS formulae, which does not change if grouping is used. Thus, parallel execution of the exponents in each group can be used to reduce the scaling of the execution time of the quantum simulation, τ, for k-local Hamiltonians to

$$\tau \in \frac{n^{k(2+o(1))-1}\,(\max_i |a_i|t)^{1+o(1)}}{\epsilon^{o(1)}}, \tag{38}$$

and for the case of physically k-local Hamiltonians it becomes

$$\tau \in \frac{n^{1+o(1)}\,(\max_i |a_i|t)^{1+o(1)}}{\epsilon^{o(1)}}, \tag{39}$$

which scales nearly-quadratically better with n than what we would expect if grouping were not used.

7. Examples

We now examine the performance of our circuit construction algorithm when applied to simulating the quantum dynamics of two important physical systems. Specifically, we examine simulating Kitaev's honeycomb model and pairing models similar to the BCS model of superconductivity.

7.1. Simulating Kitaev's honeycomb model

Consider Kitaev's honeycomb model [27] described by the Hamiltonian

$$\hat{H}^{(n)} = -J_x \sum_{x-\text{link}} X_i X_j - J_y \sum_{y-\text{link}} Y_i Y_j - J_z \sum_{z-\text{link}} Z_i Z_j,$$

with the links shown in the honeycomb-lattice representation depicted in figure 2.

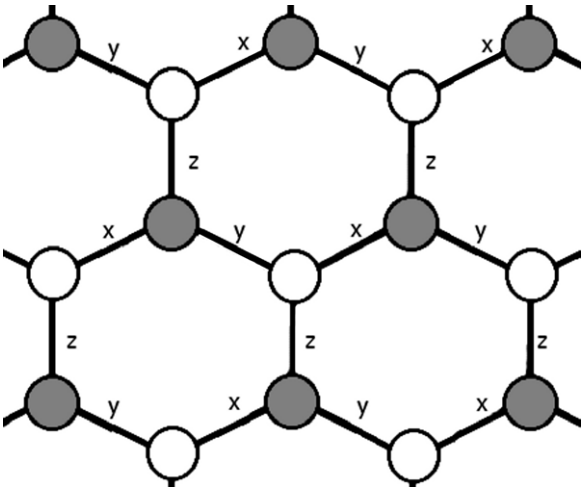

Figure 2. Kitaev's honeycomb lattice with each vertex representing a physical qubit and each edge representing an interaction between two qubits denoted by x-, y- and z-links.

Although exactly solvable [27], the ground state has applications for topological error correction and is difficult to experimentally prepare. Our simulation circuits can then be used (in conjunction with a ground-state preparation method such as adiabatic state preparation [23] or the Abrams Lloyd algorithm [19]) to prepare the ground states of such Hamiltonians.

The resulting sequence of exponentials yielded by our circuit design algorithm yields a sequence of exponentials of X_iX_j, Y_iY_j and Z_iZ_j. Figure 3 gives the simulation circuits that our algorithm uses to simulate each of these exponentials. We can use these diagrams to find the number of operations used in the simulation by using the fact that each term in the Hamiltonian appears $2(5)^{\chi-1}$ times in the TS formula and that there are $n/2$ different XX, YY and ZZ interaction terms in the Hamiltonian.

As r TS formulae are used in the simulation, the total number of times each of these three types of interactions appears is $n5^{\chi-1}r$. The total number of gates required for the simulation can be found by multiplying the number of interactions of each type by the number of gates needed to simulate that type of interactions (explicit constructions for these circuits are given in figure 3). The total number of operations required to simulate the Honeycomb model is summarized below.

- Hadamard gates: $8n5^{\chi-1}r$,
- T gates: $16n5^{\chi-1}r$,
- Z-rotation gates: $3n5^{\chi-1}r$,
- CNOT gates: $6n5^{\chi-1}r$,

where r is the number of time steps used in the simulation and χ is the iteration order of the TS formula used in the simulation. If r is chosen as per (15) then the error is promised to be less than $\epsilon/2$ given ϵ is sufficiently small [17, 18]; however, other choices of r are possible if rigorous guarantees that the simulation error is less than ϵ are not desired.

We can then directly estimate the scaling of the circuit size with the simulation parameters if χ and r are taken to be the default values for our algorithm. Kitaev's honeycomb-model

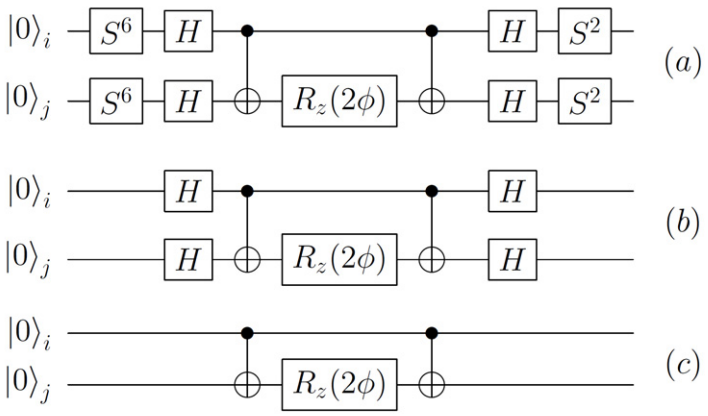

Figure 3. Circuits for simulating evolution due to exponentials of the terms present in the honeycomb model or pairing models: (a) simulates $e^{-iY_iY_j\phi}$, (b) simulates $e^{-iX_iX_j\phi}$ and (c) simulates $e^{-iZ_iZ_j\phi}$. Exponentials of the form $e^{-iZ_p\phi}$ can be trivially simulated with an R_z rotation.

Hamiltonian is physically two-local with $m \leqslant 3n/2$. Combining the observation that

$$\max_i |a_i| \leqslant \max\left\{|J_x|, |J_y|, |J_z|\right\}, \tag{40}$$

with (37) implies that our circuit-design algorithm yields $\mathcal{C}$ for simulating $U^{(n)}(t)$ within error tolerance ϵ and with a circuit size that scales as

$$N_{\text{op}} \in n^{2+o(1)} \left(\max\{|J_x|, |J_y|, |J_z|\}t\right)^{1+o(1)} / \epsilon^{o(1)}, \tag{41}$$

elementary gates from $\mathcal{G}$ acting on only n qubits. This scaling is significantly better than previous algorithms, which had a bound on the number of gates in $O(n^4 \log^* n)$ and utilize quantum oracles that may be difficult to implement [17, 18].

7.2. Simulating pairing models

Our second example simulates general pairing Hamiltonian evolution, which is central to studies of superconductivity in many-body systems. A notable example of such Hamiltonians is the BCS Hamiltonian, which describes the interaction of electrons on a lattice according to [31, 32]

$$\hat{H}_{\text{BCS}} = \frac{1}{2}\sum_{p=1}^{n} \mathcal{E}_p(a_p^\dagger a_p + a_{-p}^\dagger a_{-p}) + \sum_{p,l=1}^{n} V_{pl}\hat{a}_p^\dagger \hat{a}_{-p}^\dagger \hat{a}_l \hat{a}_{-l}, \tag{42}$$

where $\hat{a}_p$ is the fermionic annihilation operator for a fermion in states $p = (\boldsymbol{p}, \uparrow)$ and $-p = (-\boldsymbol{p}, \downarrow)$ with $\boldsymbol{p}$ the particle's momentum and $\uparrow$ and $\downarrow$ its spin, $\hat{N}_p$ is the number operator for fermions in state p, $\mathcal{E}_p$ is the on-site interaction strength and V_{pl} is the interaction strength between neighboring fermions.

The general BCS Hamiltonian (42) can be mapped to a spin system by using each spin to represent the presence or absence of a fermion in that mode. Wu *et al* [23] show that a large

class of pairing models that subsume the BCS Hamiltonian can be expressed as

$$H_p = \frac{1}{2}\sum_{p=1}^{n}\gamma_p Z_p + \sum_{r=\pm}\sum_{l>p=1}^{n} V_{pl}^{r}\left(X_p X_l + r Y_p Y_l\right),$$

for X_p, Y_p and Z_p the Pauli X, Y and Z operators applied to qubit p. Specifically, the BCS Hamiltonian has $V_{pl}^{-} = 0$ and $\gamma_p = \varepsilon_p + V_{pp}$.

As in the previous example, our circuit design algorithm yields a sequence of exponentials of the different terms in the Hamiltonian. The circuit implementations of the resulting exponentials can also be seen in figure 3. By multiplying the total number of exponentials of each type by the number of gates used in the implementation of each exponential and collecting the result, we find that our algorithm yields a circuit containing the following numbers of gates:

- Hadamard gates: $16n(n-1)5^{\chi-1}r$,
- T gates: $32n(n-1)5^{\chi-1}r$,
- Z-rotation gates: $2n(2n-1)5^{\chi-1}r$,
- CNOT gates: $8n(n-1)5^{\chi-1}r$,

where the algorithms cited scaling is found by choosing r as per equation (15) and substituting the asymptotic scaling of r for χ chosen approximately optimally [17, 18]. The value of χ that reduces the *total number* of gates can be found by minimizing the sum of these gate counts over all χ.

Previously, circuit design yielded circuits with $O(n^5 t^2)$ gates with no promises about the accuracy of the simulation [23]. The circuits yielded by our algorithm are polynomially shorter than this previous best method. This follows from the fact that H_p is two-local and hence $m \in O(n^2)$, which implies that our algorithm yields a circuit with complexity

$$N_{\mathrm{op}} \in \frac{n^{4+o(1)}\left(\max_{p,l,r}\{|\gamma_p|, |V_{pl}^{r}|\}t\right)^{1+o(1)}}{\epsilon^{o(1)}}, \tag{43}$$

when the default values of r and χ are used.

Our algorithm can thus generate efficient quantum circuits for simulating the dynamics of general BCS Hamiltonians. Our resultant circuits can be used in conjunction with eigenvector simulation techniques or adiabatic state preparation to approximate the ground state of a quantum system and can thus be useful for autonomously determining whether classes of pairing models afford exotic types of superconductivity.

8. Numerical estimates of error in low-order TS formulae for two-body Hamiltonians

The results of the previous section suggest that quantum simulations of pairing Hamiltonians may be difficult for large n because the number of required operations scales, at most, as $n^{4+o(1)}$; however, we do not know whether this upperbound is tight. Here we provide numerical evidence that the error bounds on which this scaling is based can substantially overestimate the error in the TS formula for a given simulation. The upper bound in question is used to estimate the number of time steps, r, needed in the simulation and therefore we will conclude that the upper bounds for r cited in [14, 17, 18] are too loose for some practical cases.

We analyze the TS error for Hamiltonians chosen randomly from an ensemble of two-body Hamiltonians that have their a_j independently distributed according to a Gaussian with

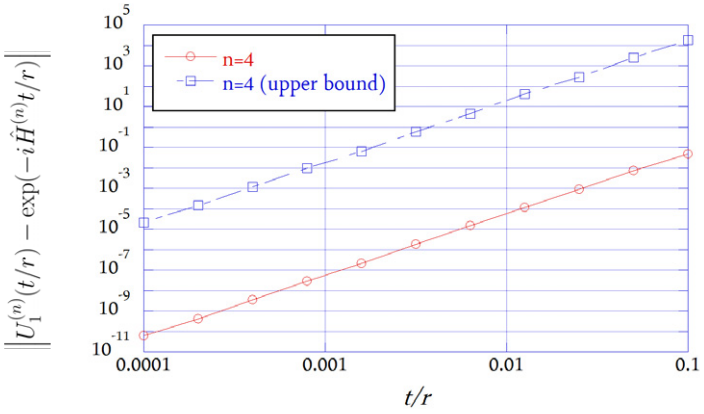

Figure 4. This plot shows the mean error or mean upper bound for the error incurred when using the lowest-order TS formula on 50 randomly generated two-body Hamiltonians acting on four qubits that were sampled from our Gaussian ensemble of Hamiltonians. This result shows that the lowest-order TS formula given in (44) can be too loose by as much as six orders of magnitude when applied to simulations of two-local Hamiltonians.

mean zero and unit variance. We choose two-body, rather than two-local Hamiltonians, for simplicity because the one-body terms present in two-local Hamiltonians become less relevant to the random Hamiltonians that we consider as n increases. Their elimination therefore allows us to estimate the scaling of the error and $\| H^{(n)} \|$ over a larger range of n.

We estimate the error invoked by using the two lowest-order TS formulae by numerically sampling random two-body Hamiltonians taken from our Gaussian ensemble. The error in the formula is measured by the two-norm of the difference between the formula and the correct exponential for the randomly generated Hamiltonian. We then repeat this for fifty different randomly generated Hamiltonians for evolutions of duration $t = 10^{-4}$ to $t = 10^{-1}$ for 2–8 qubits. Higher-order integrators can be studied similarly, but low-order integrators are often the most significant for the current generation of experiments [33]. The error bound for the Strang–splitting (denoted $U_1^{(n)}$) in [14] gives

$$\left\| U_1^{(n)}\left(\frac{t}{r}\right) - \exp\left(-i\hat{H}^{(n)}\frac{t}{r}\right)\right\| \leqslant 2\left(\frac{3\,m\,\max_i |a_i|t}{2r}\right)^3. \tag{44}$$

We find from figure 4 that the bound is, on average, too loose by approximately six orders of magnitude for $t \in [10^{-4}, 10^{-1}]$ for random two-body Hamiltonians acting on n qubits. This implies that much tighter estimates of the error in the TS formulae are needed to accurately estimate the performance of simulation algorithms.

Numerical estimates of the error are obtained by fitting the data in figure 5 that the average error for randomly sampled two-local Hamiltonians acting on 2–8 qubits is well modeled by

$$\left\| U^{(n)}\left(\frac{t}{r}\right) - U_1^{(n)}\left(\frac{t}{r}\right)\right\| \approx \frac{(\|\hat{H}^{(n)}\|\frac{t}{r})^3}{3n^2}, \tag{45}$$

for $U_1^{(n)}$ (the lowest-order TS formula). We see from the error bounds in figure 5 that the estimate of the error in $U_1(t/r)$ within an order of magnitude of the actual error for most of the randomly selected Hamiltonians.

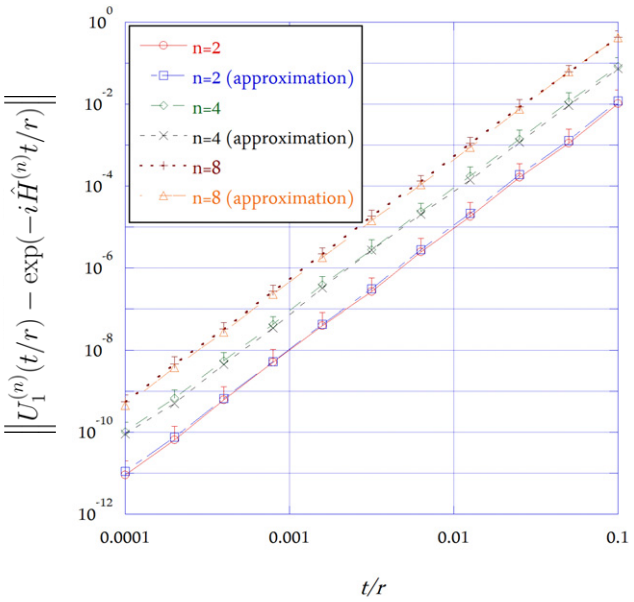

Figure 5. This plot shows the mean error in the lowest-order TS formula as a function of the duration of each time step and the approximation (45) where each point is randomly drawn from a Gaussian ensemble of two-body Hamiltonians. Error bars are computed via the standard deviations of the measured results for the ensembles and only the upper bar is plotted because the standard deviation is comparable or exceeds the mean value for all of these data points. The data are therefore within statistical error of our approximation.

Similarly, we find by fitting polynomial functions to the data in figure 6 that

$$\left\| U^{(n)}\left(\frac{t}{r}\right) - U_2^{(n)}\left(\frac{t}{r}\right) \right\| \approx \frac{1}{3000}\left(\frac{\|\hat{H}^{(n)}\|\frac{t}{r}}{\sqrt{n}}\right)^5, \tag{46}$$

for random two-local Hamiltonians. We can also find approximate forms for the error scaling for higher-order integrators, although doing so becomes more difficult as numerical precision restricts the range of t that can be used to assess the scaling.

The total number of time steps needed to approximately simulate the Hamiltonian evolution within Trotter-error $\epsilon/2$ using $U_1^{(n)}$ or $U_2^{(n)}$ follow directly from expressions (45) and (46). The simulation errors are at most additive throughout the evolution, as each operator in the product formula is unitary. This implies that the total error is at most $\epsilon/2$ for the case where $U_1^{(n)}$ is used if

$$r\left(\frac{(\|\hat{H}^{(n)}\|\frac{t}{r})^3}{3n^2}\right) \lessapprox \frac{\epsilon}{2}. \tag{47}$$

Solving for r gives the following approximate requirement for the number of time steps needed to satisfy the required tolerance

$$r \gtrapprox \sqrt{\frac{2}{3n^2\epsilon}}\left(\|\hat{H}^{(n)}\|t\right)^{1.5}. \tag{48}$$

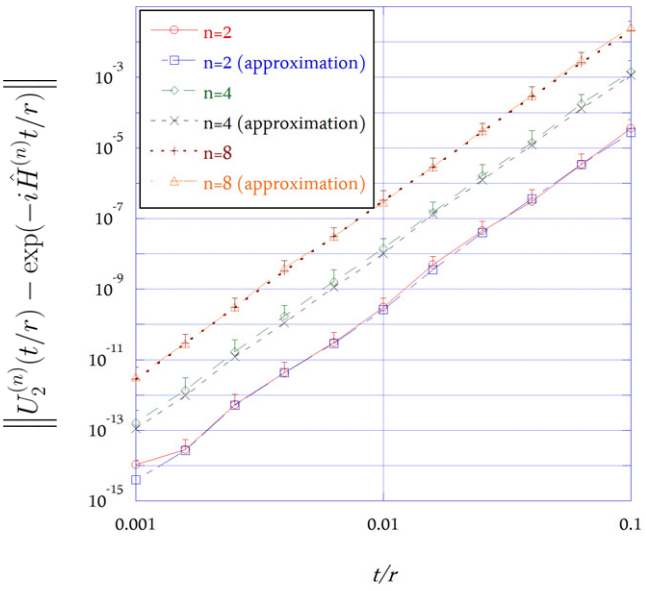

Figure 6. This plot shows the mean error in the second-lowest-order TS formula as a function of the duration of each time step and the approximation (46) for a set of 50 randomly generated two-local Hamiltonians sampled from our Gaussian ensemble for each data point. We observe that the data is within statistical error of the fit in (46).

The corresponding approximate requirement on r for $U_2^{(n)}$ is

$$r \gtrapprox \frac{1}{30}\left(\frac{540}{n^{2.5}\epsilon}\right)^{1/4}\left(\|\hat{H}^{(n)}\|t\right)^{1.25}. \tag{49}$$

These estimates of r can be used in place of the default value in algorithm 4; although we cannot rigorously guarantee that they will provide error at most ϵ.

Since there are five times as many exponentials in $U_1^{(n)}(t/r)$ than there are exponentials in $U_2^{(n)}(t/r)$, inequalities (48) and (49) suggest that $U_1^{(n)}$ provides a shorter sequence of exponentials than the sequence consisting of r $U_2^{(n)}$ formulae if

$$\sqrt{\frac{2\|\hat{H}^{(n)}\|t}{3n^2\epsilon}} \leqslant \frac{5}{30}\left(\frac{540\|\hat{H}^{(n)}\|t}{n^{2.5}\epsilon}\right)^{1/4}, \tag{50}$$

which is guaranteed if

$$\epsilon > \frac{16\|\hat{H}^{(n)}\|t}{15n^{3/2}}. \tag{51}$$

If inequality (51) is not satisfied, then higher-order formulae such as $U_2^{(n)}$ are likely to yield more efficient simulation circuits. Present simulation experiments, however, are often confined to short evolutions with relatively large error tolerances (because direct comparison with numerical results is possible). As a result, low-order approximations remain relevant for present experiments.

Tables 1 and 2 provide estimates of the number of exponentials that need to be implemented by algorithm 3 in a simulation of a random two-body Hamiltonian. We vary the number of

Table 1. Extrapolated number of exponentials of $\hat{\mathfrak{h}}_j^{(n)}$ that have to be implemented to simulate a random two-body Hamiltonian with $a_j \sim \mathcal{N}(0, 1)$ for $\epsilon = 0.01$ and $t = 0.1$.

n	N_{exp} for $U_1^{(n)}$	N_{exp} for $U_2^{(n)}$	Ratio
2	36	90	0.40
4	432	540	0.80
10	8 190	8 100	1.01
40	786 240	421 200	1.87
100	14 523 300	7573 500	1.92

Table 2. Extrapolated number of exponentials of $\hat{\mathfrak{h}}_j^{(n)}$ that have to be implemented to simulate a typical random two-body Hamiltonian with $a_j \sim \mathcal{N}(0, 1)$ for $\epsilon = 10^{-6}$ and $t = 0.01$.

n	N_{exp} for $U_1^{(n)}$	N_{exp} for $U_2^{(n)}$	Ratio
2	108	90	1.20
4	1 296	540	2.40
10	28 350	4050	7.00
40	2471 040	280 800	8.80
100	45 886 500	4455 000	10.3

qubits over an experimentally reasonable regime ($n = 2, 4$ and 10 qubits) and then extrapolate the scaling beyond the limitations of existing classical simulators to 40 and 100 qubits. In order to perform this extrapolation, we need to know the norm of the Hamiltonian. We find from data fitting for cases up to 8 qubits that the ensemble average of the norm of our random two-body Hamiltonians obeys

$$\|H^{(n)}\| \approx 1.3n^{5/3}. \tag{52}$$

Table 1 shows how the number of required exponentials varies if U_2 is used instead of U_1 for a relatively modest value of ϵ and t, whereas table 2 presents the number of exponentials for a shorter evolution with a very small value of ϵ. Note that although the time is kept constant for these data sets, the (expected) norm of the Hamiltonian does not.

The number of operations required in the simulation scales (for fixed order TS formulae) as $O(n^2 r)$. Our numerical estimate of the ensemble mean of the norm of $\|H\|$ in (52) and the approximate bounds for r in (48) and (49) lead us to the conclusion that the number of operations required to simulate a random two-body Hamiltonian scales with n as $o(n^{3.4})$ as opposed to the $n^{4+o(1)}$ scaling predicted from the upper bounds for the error incurred by using either $U_1^{(n)}(t/r)$ or $U_2^{(n)}(t/r)$. This suggests that the complexity of simulating random two-local Hamiltonians may be polynomially smaller than previous work implies.

The results in table 1 indicates that using $U_1^{(n)}(t/r)$ instead of $U_2^{(n)}(t/r)$ leads to a reduction in the simulation complexity for small values of n, but $U_2^{(n)}(t/r)$ leads to more efficient simulations for $n \geqslant 40$. In contrast, the data in table 2 shows that U_2 is more efficient

than U_1 in every case considered except for the case where $n = 2$. We can therefore conclude that low-order TS formulae can sometimes be more efficient than high-order formulae for relatively undemanding simulation problems. The data also suggests that extremely small gate errors (error on the order of 10^{-7} per gate) may be required to extend the DQS paradigm out to 40 qubits and beyond because millions of gates are expected to be required for relatively modest simulations of two-local Hamiltonians.

9. Conclusion

In conclusion, we have designed an efficient classical algorithm for autonomous construction of efficient quantum circuits to simulate state evolution. The circuits are costed in terms of a small standard gate set and require neither a Hamiltonian oracle nor ancillary qubits, thereby making the universal quantum simulator minimal in space cost. Furthermore, we show that the costing is still too pessimistic in some cases and could be improved but some orders of magnitudes. The algorithm also systematically searches out commuting terms in the Hamiltonian and groups them to reduce the depths of the circuits yielded by our algorithm; thereby reducing the time required to execute the circuits using parallelism.

Our circuit construction algorithm is a significant advance because it is straightforward to implement the algorithm on a computer, it is highly efficient and it gives upper bounds for the simulation error. Knowing error bounds is important for assessing the veracity of any quantum simulation. Our work thus provides an important step towards constructing a practical, efficient and trustworthy simulator of quantum dynamics.

There are several remaining open problems that have not been addressed by this work. First is the issue of simulating time–dependent quantum systems. Such issues can be resolved by employing TS formulae for ordered operator exponentials [14, 15] although the optimality of those error bounds for actual circuits remains to be checked using algorithmic methods we have introduced here.

Similarly, providing better upper bounds for r remains an important problem as our work has shown that in physically significant cases these bounds can be far too pessimistic. Providing such bounds would enable much more challenging simulation experiments to be performed in cases where rigorous error bounds are required of the output states.

A final important extension of this work is discussing the optimization of the resulting circuits that are yielded by the algorithm. Further optimization should be possible by using circuit identities to simplify the resulting circuits. Finding autonomous methods to optimize the output of such algorithms would be a significant asset in the development of a DQS that exceeds the power of existing classical simulators of quantum systems.

Acknowledgments

We acknowledge MITACS, USARO, NSERC and AITF for financial support and thank M Müller for helpful comments. BCS is supported by a CIFAR Fellowship.

Appendix. Condition for commutation of Pauli operators

Here we prove that (22) gives a necessary and sufficient criterion for determining whether two Hamiltonians that are tensor products of Pauli operators commute. This criterion is essential to

our discussion of parallelizing the quantum simulation because we need to divide the simulation into mutually commuting sections in order to exploit parallelism.

We simplify our discussion by removing from $\hat{\mathfrak{h}}_i^{(n)}$ and $\hat{\mathfrak{h}}_j^{(n)}$ every qubit that is either acted on as the identity operator by at least one of the Hamiltonians or every qubit that both Hamiltonians have the same action upon. These qubits are removed from consideration because they are not needed to determine the commutation properties. After removing these irrelevant qubits, we simplify the discussion by relabeling the remaining qubits to group them in six different groups. These simplifications lead to the following representation for $\hat{\mathfrak{h}}_i^{(n)}$

$$\hat{\mathfrak{h}}_i^{(n)} \sim X^{\otimes|\boldsymbol{S}_{xi}\cap\boldsymbol{S}_{yj}|}\otimes Y^{\otimes|\boldsymbol{S}_{yi}\cap\boldsymbol{S}_{zj}|}\otimes Z^{\otimes|\boldsymbol{S}_{zi}\cap\boldsymbol{S}_{xj}|} \otimes X^{\otimes|\boldsymbol{S}_{xi}\cap\boldsymbol{S}_{zj}|}\otimes Y^{\otimes|\boldsymbol{S}_{yi}\cap\boldsymbol{S}_{xj}|}\otimes Z^{\otimes|\boldsymbol{S}_{zi}\cap\boldsymbol{S}_{yj}|}. \tag{A.1}$$

The simplified expression for $\hat{\mathfrak{h}}_j^{(n)}$ is

$$\hat{\mathfrak{h}}_j^{(n)} \sim Y^{\otimes|\boldsymbol{S}_{xi}\cap\boldsymbol{S}_{yj}|}\otimes Z^{\otimes|\boldsymbol{S}_{yi}\cap\boldsymbol{S}_{zj}|}\otimes X^{\otimes|\boldsymbol{S}_{zi}\cap\boldsymbol{S}_{xj}|} \otimes Z^{\otimes|\boldsymbol{S}_{xi}\cap\boldsymbol{S}_{zj}|}\otimes X^{\otimes|\boldsymbol{S}_{yi}\cap\boldsymbol{S}_{xj}|}\otimes Y^{\otimes|\boldsymbol{S}_{zi}\cap\boldsymbol{S}_{yj}|}. \tag{A.2}$$

We can then evaluate the product of the two operators using the property that $XY = \mathrm{i}Z$, $YZ = \mathrm{i}X$ and $ZX = \mathrm{i}Y$ and also using the anti-commutativity of Pauli–operators. This implies

$$\hat{\mathfrak{h}}_i^{(n)}\hat{\mathfrak{h}}_j^{(n)} \sim (\mathrm{i}Z)^{\otimes|\boldsymbol{S}_{xi}\cap\boldsymbol{S}_{yj}|}\otimes(\mathrm{i}X)^{\otimes|\boldsymbol{S}_{yi}\cap\boldsymbol{S}_{zj}|}\otimes(\mathrm{i}Y)^{\otimes|\boldsymbol{S}_{zi}\cap\boldsymbol{S}_{xj}|} \otimes(-\mathrm{i}Y)^{\otimes|\boldsymbol{S}_{xi}\cap\boldsymbol{S}_{zj}|}\otimes(-\mathrm{i}Z)^{\otimes|\boldsymbol{S}_{yi}\cap\boldsymbol{S}_{xj}|}\otimes(-\mathrm{i}X)^{\otimes|\boldsymbol{S}_{zi}\cap\boldsymbol{S}_{yj}|}. \tag{A.3}$$

Conversely,

$$\hat{\mathfrak{h}}_j^{(n)}\hat{\mathfrak{h}}_i^{(n)} \sim (-\mathrm{i}Z)^{\otimes|\boldsymbol{S}_{xi}\cap\boldsymbol{S}_{yj}|}\otimes(-\mathrm{i}X)^{\otimes|\boldsymbol{S}_{yi}\cap\boldsymbol{S}_{zj}|}\otimes(-\mathrm{i}Y)^{\otimes|\boldsymbol{S}_{zi}\cap\boldsymbol{S}_{xj}|} \otimes(\mathrm{i}Y)^{\otimes|\boldsymbol{S}_{xi}\cap\boldsymbol{S}_{zj}|}\otimes(\mathrm{i}Z)^{\otimes|\boldsymbol{S}_{yi}\cap\boldsymbol{S}_{xj}|}\otimes(\mathrm{i}X)^{\otimes|\boldsymbol{S}_{zi}\cap\boldsymbol{S}_{yj}|}. \tag{A.4}$$

We then see from (A.3) and (A.4) that $[\hat{\mathfrak{h}}_i^{(n)}, \hat{\mathfrak{h}}_j^{(n)}] = 0$ if and only if

$$|\boldsymbol{S}_{y_i}\cap\boldsymbol{S}_{x_j}| + |\boldsymbol{S}_{x_i}\cap\boldsymbol{S}_{z_j}| + |\boldsymbol{S}_{z_i}\cap\boldsymbol{S}_{x_y}| \equiv |\boldsymbol{S}_{x_i}\cap\boldsymbol{S}_{y_j}| + |\boldsymbol{S}_{z_i}\cap\boldsymbol{S}_{x_j}| + |\boldsymbol{S}_{y_i}\cap\boldsymbol{S}_{z_y}| \pmod 2, \tag{A.5}$$

which is equivalent to (22).

References

[1] Feynman R P 1982 *Int. J. Theor. Phys.* **21** 467
[2] Lloyd S 1996 *Science* **273** 1073
[3] Asupuru-Guzik A, Dutoi A D, Love P J and Head-Gordon M 2005 *Science* **309** 1704
[4] Weimer H, Müller M, Lesanovsky I, Zoller P and Bühler H-P 2010 *Nature Phys.* **6** 382
[5] Weimer H, Müller M, Bühler H-P and Lesanovsky I 2011 *Quantum Inform. Process.* **10** 885
[6] Whitfield J D, Biamonte J and Asupuru-Guzik A 2011 *Mol. Phys.* **109** 735
[7] Lanyon B P *et al* 2011 *Science* **334** 57
[8] Duan L-M, Demler E and Lukin M D 2003 *Phys. Rev. Lett.* **91** 090402
[9] Aguado M, Brennen G K, Verstraete F and Cirac J I 2008 *Phys. Rev. Lett.* **101** 260501
[10] Gerritsma R, Kirchmair G, Zähringer F, Solano E, Blatt R and Roos C-F 2009 *Nature* **463** 68
[11] Buluta I and Nori F 2009 *Science* **326** 108

[12] Friedenauer A, Schmitz H, Glückert J, Porras D and Schätz T 2008 *Nature Phys.* **4** 757
[13] Kitaev A, Shen A H and Vyalyi M N 2002 *Classical and Quantum Computation* (*Graduate Studies in Mathematics* vol 47) (Providence, RI: American Mathematical Society)
[14] Wiebe N, Berry D W, Høyer P and Sanders B C 2010 *J. Phys. A: Math. Theor.* **43** 065203
[15] Wiebe N, Berry D W, Høyer P and Sanders B C 2011 *J. Phys. A: Math. Theor.* **44** 445308
[16] Papageorgiou A and Zhang C 2012 *Quantum Inform. Process.* **11** 541
[17] Berry D W, Ahokas G, Cleve R and Sanders B C 2007 *Commun. Math. Phys.* **270** 359
[18] Berry D W, Ahokas G, Cleve R and Sanders B C 2007 Mathematics of quantum computation and quantum technology *Quantum Algorithms for Hamiltonian Simulation* ed G Chen, L H Kauffman and S J Lomonaco (Oxford: Chapman and Hall) chapter 4, pp 89–110
[19] Abrams D S and Lloyd S 1997 *Phys. Rev. Lett.* **79** 2586
[20] Jordan S P, Lee K S M and Preskill J 2011 arXiv:1111.3633
[21] Jordan S P, Lee K S M and Preskill J 2011 arXiv:1112.4833
[22] Hohenberg P and Kohn W 1964 *Phys. Rev.* **136** B864
[23] Wu L-A, Byrd M S and Lidar D A 2002 *Phys. Rev. Lett.* **89** 057904
[24] Suzuki M 1990 *Phys. Lett.* **146** 319
[25] Suzuki M 1991 *J. Math. Phys.* **32** 400
[26] Nielsen M A and Chuang I L 2000 *Quantum Computation and Quantum Information* (Cambridge: Cambridge University Press)
[27] Kitaev A Y 2006 *Ann. Phys.* **321** 2
[28] Dawson C M and Nielsen M A 2006 *Quantum Inform. Process.* **6** 81
[29] Childs A M and Kothari R 2010 *Quantum Inform. Comput.* **10** 669
[30] Vizing V G 1964 *Diskret. Analiz.* **3** 25
[31] Ring P and Schuck P 1980 *The Nuclear Many-Body Problem* (Heidelberg: Springer)
[32] Mahan G D 2000 *Many-Particle Physics* 3rd edn (New York: Kluwer–Plenum)
[33] Brown K R, Clark R J and Chuang I L 2006 *Phys. Rev. Lett.* **97** 050504