

Lawrence Berkeley National Laboratory

LBL Publications

Title

An Algebraic Quantum Circuit Compression Algorithm for Hamiltonian Simulation

Permalink

<https://escholarship.org/uc/item/47z4n8n6>

Authors

Camps, Daan
Kökcü, Efehan
Bassman, Lindsay
[et al.](#)

Publication Date

2021-08-06

AN ALGEBRAIC QUANTUM CIRCUIT COMPRESSION ALGORITHM FOR HAMILTONIAN SIMULATION*

DAAN CAMPS[†], EFEKAN KÖKCÜ[‡], LINDSAY BASSMAN[†], WIBE A. DE JONG[†],
ALEXANDER F. KEMPER[‡], AND ROEL VAN BEEUMEN[†]

Abstract. Quantum computing is a promising technology that harnesses the peculiarities of quantum mechanics to deliver computational speedups for some problems that are intractable to solve on a classical computer. Current generation noisy intermediate-scale quantum (NISQ) computers are severely limited in terms of chip size and error rates. Shallow quantum circuits with uncomplicated topologies are essential for successful applications in the NISQ era. Based on matrix analysis, we derive localized circuit transformations to efficiently compress quantum circuits for simulation of certain spin Hamiltonians known as free fermions. The depth of the compressed circuits is independent of simulation time and grows linearly with the number of spins. The proposed numerical circuit compression algorithm behaves backward stable and scales cubically in the number of spins enabling circuit synthesis beyond $\mathcal{O}(10^3)$ spins. The resulting quantum circuits have a simple nearest-neighbor topology, which makes them ideally suited for NISQ devices.

Key words. quantum computing, quantum circuit synthesis, algebraic circuit compression, Hamiltonian simulation, free fermions, NISQ

AMS subject classifications. 15A23, 15A69, 65Z05, 68Q12, 81P65, 81R12

1. Introduction. The field of quantum computing [28] is rapidly evolving. The current generation of quantum hardware is known as *Noisy Intermediate-Scale Quantum* (NISQ) computers [30] and can perform specialized computational tasks that become rapidly intractable for a classical computer [1]. A computational task for a quantum computer, or *quantum program*, is typically expressed as a *quantum circuit* that consists of a sequence of unitary transformations [28]. Each of these unitary transformations typically acts on just one or two *qubits* of the quantum computer and they are often referred to as *quantum gates*. The technological limitations in NISQ hardware impose substantial constraints both on the number of qubits and on the number of unitary operations, also known as *circuit depth*, that can be performed. Noise introduced by two qubit gates eventually reduces the fidelity of the quantum state until a useful signal can no longer be measured. Shallow and simple quantum circuit structures are thus crucial for successful applications in the NISQ era.

Quantum circuit *compilation* or *synthesis* [4, 31] is the problem of computing a circuit representation into two qubit operations for a target unitary matrix. General purpose synthesis algorithms based on well-known matrix decompositions have been proposed in the literature, for example, based on Givens [41] or Householder [18] QR factorization of the target unitary. A more efficient algebraic synthesis algorithm in terms of circuit complexity is known as the *quantum Shannon decomposition* [31]

*Submitted to the editors August 19, 2021.

Funding: DC and RVB are supported by the Laboratory Directed Research and Development Program of Lawrence Berkeley National Laboratory under U.S. Department of Energy Contract No. DE-AC02-05CH11231. LB, and WAdJ were supported by the U.S. Department of Energy (DOE) under Contract No. DE-AC02-05CH11231, through the Office of Advanced Scientific Computing Research Accelerated Research for Quantum Computing Program. EK, and AFK were supported by the Department of Energy, Office of Basic Energy Sciences, Division of Materials Sciences and Engineering under Grant No. DE-SC0019469.

[†]Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States. (dcamps@lbl.gov, lbassman@lbl.gov, wadejong@lbl.gov, rvanbeeumen@lbl.gov).

[‡]Department of Physics, North Carolina State University, Raleigh, NC 27695, United States. (ekokcu@ncsu.edu, akemper@ncsu.edu).

and is based on a hierarchical cosine-sine decomposition (CSD) [32,33] of the unitary matrix. While these methods work for every unitary matrix, they have two major disadvantages. Firstly, they require an exorbitant amount of classical resources. The dimension of the unitary matrix for N qubits to be decomposed is $2^N \times 2^N$. Storing this matrix on a classical computer rapidly becomes intractable, let alone computing a decomposition of cubic complexity in the matrix dimension such as a QR factorization or CSD. Secondly, the circuits that are derived from these synthesis algorithms contain, in general, exponentially many gates in terms of the number of qubits. For many applications of practical interest, more efficient circuits can be obtained with optimization methods [42] or by exploiting certain structures in the unitary [11]. This approach still suffers from the first issue as the full unitary has to be formed.

In this paper we propose an application-specific circuit compression and synthesis algorithm that overcomes both challenges. We never form the $2^N \times 2^N$ unitary and the compression algorithm has a cubic complexity in N which is an exponential improvement compared to a cubic dependence on 2^N [18,31,41]. Furthermore, the compressed circuits have a simple nearest-neighbor topology, a circuit depth of $\mathcal{O}(N)$, and $\mathcal{O}(N^2)$ quantum gates. This makes them ideally suited for the NISQ era and in particular for hardware based on superconducting qubits. The application that our synthesis algorithm is designed for is known as *Hamiltonian simulation* [24] which involves the evolution of a quantum state of the system under the time-dependent Schrödinger equation. This problem is ubiquitous in quantum chemistry [6,8] and physics, for example in adiabatic ground state preparation [5]. We show that quantum circuits for the time evolution of certain spin models, known in physics as free fermionizable or integrable models, are efficiently compressible. Our analysis leads to an algebraic circuit compression algorithm that behaves as a backward stable algorithm in practice. MATLAB and C++ implementations of our algorithms are publicly available as part of the *fast free fermion compiler* (F3C) [12,37] at <https://github.com/QuantumComputingLab>. F3C is build based on the QCLAB toolbox [13,38] for creating and representing quantum circuits.

A related algorithm based on a Givens QR factorization of an $N \times N$ matrix formed by the quadratic Hamiltonian was proposed in [21] to generate Slater determinants. This method was further developed to prepare a Hartree-Fock wave function [2] and generic fermionic Gaussian states [19]. It assumes that the full circuit maps to a free fermionic system, while our localized operations can still be used for circuits that are partially comprised of specific quantum gates.

This paper is accompanied by a dual paper targeted at the physics community [22] that analyzes the properties of the Hamiltonians from studying the Hamiltonian algebra. While this paper focuses on the matrix structures and the efficient and accurate numerical computation of the compression, [22] focuses on the implications for the physics community and showcases the results of an adiabatic state preparation experiment performed on quantum hardware that is only feasible due to the compressed quantum circuits.

The remainder of our paper is organized as follows. [Section 2](#) provides a more detailed introduction to the problem of Hamiltonian simulation, reviews the concept of operator splitting methods to solve this problem, introduces the specific spin Hamiltonians for which our compression algorithm works, and relates the current paper to earlier work. [Section 3](#) provides an overview of useful elementary results on Pauli rotation matrices and parameterizations of $SU(2)$ that we will use for the remainder of the analysis. [Section 4](#) shows that compressing quantum circuits for the simulation of classical Ising models to depth $\mathcal{O}(1)$ immediately follows from [Section 3](#). We pres-

ent our circuit compression algorithms for simulation circuits that are comprised of gates that allow for a *fusion* and *turnover* operation in Section 5. Section 6 demonstrates based on the results from Section 3 that Kitaev chains and XY Hamiltonians satisfy these criteria and can be efficiently compressed. Section 7 shows the same for transverse-field XY Hamiltonians and the special case of transverse-field Ising models. Details of the implementation and considerations on numerical stability are provided in Section 8. Section 9 provides numerical examples that demonstrate the speed and accuracy of our method. We conclude in Section 10.

2. Problem statement and preliminary results. In this section we review the problem statement and preliminary results about Hamiltonian simulation on quantum computers.

2.1. Hamiltonian simulation. Simulating a quantum system of N spins or qubits involves the evolution of the quantum state of the system under the Schrödinger equation,

$$\frac{\partial}{\partial t}\psi(t) = -iH(t)\psi(t),$$

and is fully determined by $H(t) \in \mathbb{C}^{2^N \times 2^N}$, the time-dependent Hamiltonian of the system, and the initial state of the system, $\psi(t_0) \in \mathbb{C}^{2^N}$. The Hamiltonian is a time-dependent Hermitian operator of exponential dimension in the system size and the quantum state $\psi(t)$ is a vector of unit norm.

Simulating from initial time t_0 to final time t_1 is achieved by the time-evolution operator

$$(2.1) \quad U(t_1, t_0) = \mathcal{T} \exp \left(-i \int_{t_0}^{t_1} H(t) dt \right),$$

where $\mathcal{T} \exp$ is the *time-ordered matrix exponential*. The final state at time t_1 becomes $\psi(t_1) = U(t_1, t_0)\psi(t_0)$. For a time-independent Hamiltonian the closed-form solution is $U(t_1, t_0) = \exp(-i(t_1 - t_0)H)$. Time evolution is a hard problem to solve on a classical computer due to the exponential dimensionality of the state space and the time-dependence of the Hamiltonian. In *digital quantum simulation*, (2.1) is evaluated on a quantum computer which naturally operates in a state space of exponential dimension. Multiple quantum algorithms have been proposed with (near) optimal asymptotic scaling [9, 10, 15, 16, 20, 25, 26]. All of these algorithms rely on more complicated circuit structures that are not well-suited for the constraints imposed by NISQ computers where the circuit depth is limited. Quantum circuits derived from *operator splitting* [27, 35] methods, also known as *Trotter product formulas* [14, 17, 34, 36] in the physics community, often lead to simple circuit structures but with a circuit depth that usually depends linearly on simulation time.

2.2. Operator splitting methods. We rely on two approximations in order to implement (2.1) on a quantum computer using an operator splitting method. First, we discretize in time by approximating $H(t)$ by a piecewise constant function with n_t time-steps of length Δt that discretize the interval $[t_0, t_1]$ [29]:

$$H(t) \approx H(t_\tau) =: H_\tau, \quad 0 < \tau \leq n_t, \quad t_\tau = t_0 + \tau\Delta t, \quad t \in [t_{\tau-1}, t_\tau).$$

Second, we approximate the matrix exponential of H_τ by products of matrix exponentials that are easier to implement on a quantum computer. The simplest case

is a first-order product formula which decomposes the Hamiltonian operator in a sum of two terms $H = A + B$. The approximate time-evolution operator for time-step Δt , $U(\Delta t) = \exp(-iA\Delta t)\exp(-iB\Delta t)$, satisfies [14]:

$$\|U(\Delta t) - \exp(-iH\Delta t)\| \leq \frac{\Delta t^2}{2} \|[A, B]\|,$$

where $[A, B] := AB - BA$. This result can be bootstrapped to show that for $H = \sum_{\ell} H_{\ell}$ and $U(\Delta t) = \prod_{\ell} \exp(-iH_{\ell}\Delta t)$, we have that

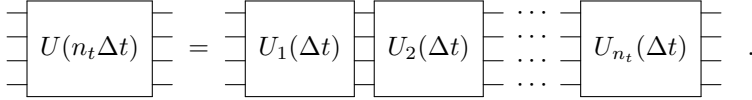
$$(2.2) \quad \|U(\Delta t) - \exp(-iH\Delta t)\| \leq \frac{\Delta t^2}{2} \sum_{i>j} \|[H_i, H_j]\|.$$

Without loss of generality, we will only use first-order Trotter decompositions throughout this paper.

Combining the discretization in time ($H(t) \approx H_{\tau}$) and the Trotter decomposition of the Hamiltonian, $H_{\tau} = \sum_{\ell} H_{\ell, \tau}$, we have the following approximation to the time-evolution operator

$$(2.3) \quad U(n_t \Delta t) = \prod_{k=0}^{n_t-1} U_{n_t-k}(\Delta t), \quad U_{\tau}(\Delta t) = \prod_{\ell} \exp(-iH_{\ell, \tau} \Delta t),$$

where the index τ is a time-ordered multiplication over n_t discretized time-steps Δt and ℓ multiplies over the terms in the Trotter decomposition. Quantum circuits based on this formula naturally become a concatenation of blocks that implement the individual time-steps and their depth grows linearly with n_t :



Note that in quantum circuit diagrams, time flows from left to right, which means that the order of operations is reversed compared to (2.3).

2.3. Spin Hamiltonians. Our compression results only hold for certain time-dependent, *ordered* or *disordered* Hamiltonians that model chains of spin-1/2 particles with a nearest-neighbor coupling and external magnetic field. These Hamiltonians are typically expressed in terms of the Pauli spin-1/2 matrices,

$$\sigma^x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma^y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma^z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

which are generators for the group of 2×2 unitary matrices with unit determinant, also known as $SU(2)$. We often write σ^{α} where $\alpha \in \{x, y, z\}$ as many of our results are independent of the type of Pauli matrix. A basis for the Hilbert space of composite quantum systems is constructed through the tensor product of the state spaces of the individual systems. To this end it is useful to introduce an abbreviated notation for a σ^{α} matrix that acts on the i th spin in a chain of N spins:

$$(2.4) \quad \sigma_i^{\alpha} := \underbrace{I \otimes \cdots \otimes I}_{i-1} \otimes \sigma^{\alpha} \otimes \underbrace{I \otimes \cdots \otimes I}_{N-i},$$

where I is the 2×2 identity matrix.

The most complicated class of Hamiltonians that we consider is known as the *time-dependent, disordered transverse field XY* (TFXY) model. The Hamiltonian is given by

$$(2.5) \quad H(t) = \underbrace{\sum_{i=1}^{N-1} J_i^x(t) \sigma_i^x \sigma_{i+1}^x + J_i^y(t) \sigma_i^y \sigma_{i+1}^y}_{\text{Coupling}} + \underbrace{\sum_{i=1}^N h_i^z(t) \sigma_i^z}_{\text{External Field}}.$$

The other two permutations of x , y , and z result in TFXZ and TFYZ Hamiltonians for which our circuit compression method also works.

The parameters J^x , J^y , and h^z in the Hamiltonian (2.5) depend both on time t and on the index i of the spin-1/2 particle in the chain. The dependence on the index i means that the Hamiltonian is disordered. If the parameters are independent of i , the Hamiltonian is called ordered. We discuss TFXY Hamiltonians in detail in Section 7. Other models that are compressible are classical Ising models, Kitaev chains, XY models, and transverse-field Ising models (TFIM). All of these are subclasses of TFXY Hamiltonians obtained by restricting some parameters of the full TFXY model.

2.4. Related work. Besides [22], this paper is closely related to two earlier papers [7, 23] written by some of us. The results in our current paper were first conjectured in [7]. There, the fixed-depth circuit property was identified by using QFAST [42], an optimization-based numerical circuit compiler. This compilation method becomes challenging for problems larger than 7 qubits because of the exponential dimensionality of the state space and this prompted us to analyze the problem in more detail. Our analysis presented in the current paper resulted in a constructive proof of the fixed-depth property (Section 5) for all cases conjectured in [7] and a scalable and accurate circuit compression algorithm that easily handles systems with $\mathcal{O}(10^3)$ qubits. Furthermore, we improve the circuit depth of the classical Ising model to $\mathcal{O}(1)$ compared to [7]. In [23] the existence of fixed depth circuits for Hamiltonian simulation is proven through a Cartan decomposition of the Lie algebra generated by the Hamiltonian. The advantage of our current approach over [23] is that by compressing an easy to generate Trotter circuit to shallow depth, we avoid having to optimize the whole circuit at once, which again only scales up to 10 qubits [23]. Furthermore, our results are derived from matrix analysis and our algorithms are exact up to machine precision.

3. Elementary properties and results. We give an overview of all properties that we use later in the analysis of the circuit compression algorithm. We start with the following well-known commutation relations between Pauli matrices:

$$(3.1) \quad [\sigma^\alpha, \sigma^\beta] = 2i\varepsilon_{\alpha\beta\gamma}\sigma^\gamma, \quad \{\alpha, \beta, \gamma\} \subseteq \{x, y, z\},$$

with $\varepsilon_{\alpha\beta\gamma}$ the Levi-Cevita tensor. Kronecker products of two Pauli matrices do commute:

$$(3.2) \quad [\sigma^\alpha \otimes \sigma^\alpha, \sigma^\beta \otimes \sigma^\beta] = 0, \quad \alpha, \beta \in \{x, y, z\}.$$

DEFINITION 3.1. For $\alpha \in \{x, y, z\}$, we define a single-spin Pauli- α rotation over an angle θ as

$$(3.3) \quad R^\alpha(\theta) := \exp(-i\sigma^\alpha\theta/2) = \text{---} \boxed{\alpha} \text{---}.$$

If the Pauli- α rotation acts on the i th spin, we denote it as $R_i^\alpha(\theta)$. Similarly, a two-spin Pauli- α rotation over an angle θ is defined as

$$(3.4) \quad R^{\alpha\alpha}(\theta) := \exp(-i\sigma^\alpha \otimes \sigma^\alpha \theta/2) = \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \alpha \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array},$$

and denoted as $R_i^{\alpha\alpha}(\theta)$ if it acts on the nearest-neighbor spins $i, i+1$.

We introduced our diagrammatic notation for single- and two-spin Pauli-rotations in [Definition 3.1](#). This will be our quantum circuit representation for these unitary matrices. The vertical direction indicates the spins on which the unitary operations are performed. The matrix representation of the single- and two-spin Pauli rotations are given by:

$$\begin{aligned} R^x(\theta) &= \begin{bmatrix} c & -is \\ -is & c \end{bmatrix}, & R^{xx}(\theta) &= \begin{bmatrix} c & & & -is \\ & c & -is & \\ & -is & c & \\ -is & & & c \end{bmatrix}, \\ R^y(\theta) &= \begin{bmatrix} c & -s \\ s & c \end{bmatrix}, & R^{yy}(\theta) &= \begin{bmatrix} c & & & is \\ & c & -is & \\ & -is & c & \\ is & & & c \end{bmatrix}, \\ R^z(\theta) &= \begin{bmatrix} e^{-i\theta/2} & & & \\ & e^{i\theta/2} & & \\ & & e^{i\theta/2} & \\ & & & e^{-i\theta/2} \end{bmatrix}, & R^{zz}(\theta) &= \begin{bmatrix} e^{-i\theta/2} & & & \\ & e^{i\theta/2} & & \\ & & e^{i\theta/2} & \\ & & & e^{-i\theta/2} \end{bmatrix}, \end{aligned}$$

where $c = \cos(\theta/2)$ and $s = \sin(\theta/2)$.

A property that we often use implicitly is the mixed product property of the Kronecker product and the observation that the identity commutes with every matrix:

$$(3.5) \quad (A \otimes I)(I \otimes B) = (I \otimes B)(A \otimes I), \quad \begin{array}{c} \text{---} \textcircled{A} \text{---} \\ | \\ \text{---} \textcircled{B} \text{---} \end{array} = \begin{array}{c} \text{---} \textcircled{A} \text{---} \\ | \\ \text{---} \textcircled{B} \text{---} \end{array}$$

Here we have a first illustration of the reversed order of operations in matrix notation compared to the schematic notation.

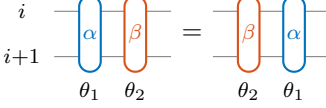
The following three lemmas list useful properties of the Pauli rotation matrices that are directly verified from [Definition 3.1](#) in combination with the commutation relations [\(3.1\)](#), [\(3.2\)](#), and [\(3.5\)](#). The first lemma provides some useful commutation relations for Pauli rotation matrices.

LEMMA 3.2. *For $\alpha, \beta \in \{x, y, z\}$, $i \in \{1, \dots, N-1\}$, the following commutation relations hold for the Pauli rotations:*

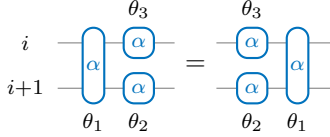
(i) *Two-spin rotations of the same type on overlapping spins:*

$$R_{i+1}^{\alpha\alpha}(\theta_2) R_i^{\alpha\alpha}(\theta_1) = R_i^{\alpha\alpha}(\theta_1) R_{i+1}^{\alpha\alpha}(\theta_2), \quad \begin{array}{c} i \\ | \\ i+1 \\ | \\ i+2 \end{array} \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \alpha \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \alpha \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \alpha \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \alpha \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array}$$

(ii) Two-spin rotations of different type on the same spins:

$$R_i^{\beta\beta}(\theta_2)R_i^{\alpha\alpha}(\theta_1) = R_i^{\alpha\alpha}(\theta_1)R_i^{\beta\beta}(\theta_2),$$


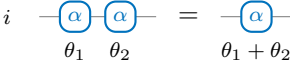
(iii) Single- and two-spin rotations of the same type on the same spins:

$$R_i^{\alpha}(\theta_3)R_{i+1}^{\alpha}(\theta_2)R_i^{\alpha\alpha}(\theta_1) = R_i^{\alpha\alpha}(\theta_1)R_{i+1}^{\alpha}(\theta_2)R_i^{\alpha}(\theta_3),$$


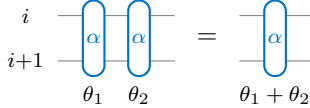
The next lemma shows that rotations of the same type acting on the same spins can be easily fused together.

LEMMA 3.3. Let $\alpha \in \{x, y, z\}$.

(i) For $i \in \{1, \dots, N\}$, single-spin Pauli- α rotations acting on the same spins can be fused or multiplied together:

$$R_i^{\alpha}(\theta_2)R_i^{\alpha}(\theta_1) = R_i^{\alpha}(\theta_1 + \theta_2),$$


(ii) For $i \in \{1, \dots, N-1\}$, two-spin Pauli- α rotations acting on the same spins can be fused or multiplied together:

$$R_i^{\alpha\alpha}(\theta_2)R_i^{\alpha\alpha}(\theta_1) = R_i^{\alpha\alpha}(\theta_1 + \theta_2),$$


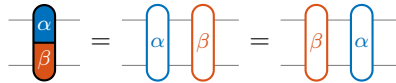
The following result directly follows from the commutativity (3.2).

LEMMA 3.4. For $\alpha, \beta \in \{x, y, z\}$, $\alpha \neq \beta$, $0 \leq \theta_\alpha, \theta_\beta < 4\pi$, we have that

$$\exp(-i(\sigma^\alpha \otimes \sigma^\alpha \theta_\alpha/2 + \sigma^\beta \otimes \sigma^\beta \theta_\beta/2)) = \exp(-i\sigma^\alpha \otimes \sigma^\alpha \theta_\alpha/2) \exp(-i\sigma^\beta \otimes \sigma^\beta \theta_\beta/2),$$

$$R^{\alpha\beta}(\theta_\alpha, \theta_\beta) = R^{\alpha\alpha}(\theta_\alpha)R^{\beta\beta}(\theta_\beta),$$

or as a circuit diagram:



3.1. Euler decompositions of SU(2). The group SU(2) is given by

$$(3.6) \quad \text{SU}(2) = \left\{ \begin{bmatrix} \alpha & -\bar{\beta} \\ \beta & \bar{\alpha} \end{bmatrix} : \alpha, \beta \in \mathbb{C}, |\alpha|^2 + |\beta|^2 = 1 \right\},$$

and it is well-known (see for example [28, Theorem 4.1]) that any element of SU(2) can be parametrized by three *Euler angles*. We will refer to this as an *Euler decomposition* of SU(2).

LEMMA 3.5. Let $\alpha, \beta \in \{x, y, z\}$, $\alpha \neq \beta$. Every matrix $U \in \text{SU}(2)$ can be represented as:

$$(3.7) \quad U = R^\alpha(\theta_1) R^\beta(\theta_2) R^\alpha(\theta_3), \quad \text{---} \begin{array}{c} \textcircled{\alpha} \textcircled{\beta} \textcircled{\alpha} \\ \theta_3 \quad \theta_2 \quad \theta_1 \end{array} \text{---}.$$

The decomposition is unique, except for a set of measure 0, if the angles are restricted to $0 \leq \theta_1 < 2\pi$, $0 \leq \theta_2 \leq \pi$ and $0 \leq \theta_3 < 4\pi$.

Proof. We give a proof for $\alpha = z$, $\beta = y$, other cases follow from a similar argument. Direct computation yields:

$$(3.8) \quad \begin{aligned} R^z(\theta_1) R^y(\theta_2) R^z(\theta_3) &= \begin{bmatrix} e^{-i\theta_1/2} & \\ & e^{i\theta_1/2} \end{bmatrix} \begin{bmatrix} \cos(\theta_2/2) & -\sin(\theta_2/2) \\ \sin(\theta_2/2) & \cos(\theta_2/2) \end{bmatrix} \begin{bmatrix} e^{-i\theta_3/2} & \\ & e^{i\theta_3/2} \end{bmatrix}, \\ &= \begin{bmatrix} \cos(\theta_2/2)e^{-i(\theta_1+\theta_3)/2} & -\sin(\theta_2/2)e^{-i(\theta_1-\theta_3)/2} \\ \sin(\theta_2/2)e^{i(\theta_1-\theta_3)/2} & \cos(\theta_2/2)e^{i(\theta_1+\theta_3)/2} \end{bmatrix}. \end{aligned}$$

It is clear that this parametrizes $\text{SU}(2)$ and that the Euler angles are unique unless $\theta_2 = 0, \pi$. \square

We can *turn over* an Euler decomposition of $\text{SU}(2)$ to its dual decomposition as shown in the following result.

LEMMA 3.6. Let $\alpha, \beta \in \{x, y, z\}$, $\alpha \neq \beta$. For every set of Euler angles given by $\theta_1, \theta_2, \theta_3$ there exists a set of dual Euler angles given by $\theta_a, \theta_b, \theta_c$ such that,

$$R^\alpha(\theta_1) R^\beta(\theta_2) R^\alpha(\theta_3) = R^\beta(\theta_a) R^\alpha(\theta_b) R^\beta(\theta_c), \quad \text{---} \begin{array}{c} \textcircled{\alpha} \textcircled{\beta} \textcircled{\alpha} \\ \theta_3 \quad \theta_2 \quad \theta_1 \end{array} \text{---} = \text{---} \begin{array}{c} \textcircled{\beta} \textcircled{\alpha} \textcircled{\beta} \\ \theta_c \quad \theta_b \quad \theta_a \end{array} \text{---}.$$

The relation between both sets of Euler angles is given by:

$$(3.9) \quad \begin{aligned} \tan\left(\frac{\theta_a + \theta_c}{2}\right) &= \tan\left(\frac{\theta_2}{2}\right) \frac{\cos((\theta_1 - \theta_3)/2)}{\cos((\theta_1 + \theta_3)/2)}, \\ \tan\left(\frac{\theta_a - \theta_c}{2}\right) &= -\tan\left(\frac{\theta_2}{2}\right) \frac{\sin((\theta_1 - \theta_3)/2)}{\sin((\theta_1 + \theta_3)/2)}, \end{aligned}$$

and

$$(3.10) \quad \begin{aligned} \tan\left(\frac{\theta_1 + \theta_3}{2}\right) &= \tan\left(\frac{\theta_b}{2}\right) \frac{\cos((\theta_a - \theta_c)/2)}{\cos((\theta_a + \theta_c)/2)}, \\ \tan\left(\frac{\theta_1 - \theta_3}{2}\right) &= -\tan\left(\frac{\theta_b}{2}\right) \frac{\sin((\theta_a - \theta_c)/2)}{\sin((\theta_a + \theta_c)/2)}. \end{aligned}$$

Proof. We again only give the proof for $\alpha = z$, $\beta = y$, but the result holds in general. In this case, the left-hand side of Lemma 3.6 is given by (3.8). The right-hand side is equal to:

$$\begin{bmatrix} \cos(\theta_a/2) & -\sin(\theta_a/2) \\ \sin(\theta_a/2) & \cos(\theta_a/2) \end{bmatrix} \begin{bmatrix} e^{-i\theta_b/2} & \\ & e^{i\theta_b/2} \end{bmatrix} \begin{bmatrix} \cos(\theta_c/2) & -\sin(\theta_c/2) \\ \sin(\theta_c/2) & \cos(\theta_c/2) \end{bmatrix}.$$

As this is an $\text{SU}(2)$ matrix, it is determined by its first column. After performing the matrix products and using some elementary trigonometry relations, we find the first column to be:

$$(3.11) \quad \begin{aligned} (1, 1) &: \cos(\theta_b/2) \cos((\theta_a + \theta_c)/2) - i \sin(\theta_b/2) \cos((\theta_a - \theta_c)/2), \\ (2, 1) &: \cos(\theta_b/2) \sin((\theta_a + \theta_c)/2) - i \sin(\theta_b/2) \sin((\theta_a - \theta_c)/2). \end{aligned}$$

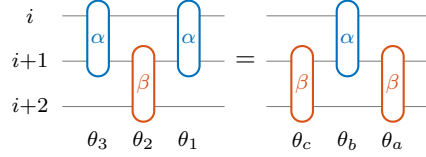
The lemma directly follows from setting the first column of this matrix to the first column of (3.8). \square

The following two lemmas are closely related to Lemma 3.6.

LEMMA 3.7. *Let $\alpha, \beta \in \{x, y, z\}$, $\alpha \neq \beta$, $i \in \{1, \dots, N-1\}$. For every set of Euler angles $\theta_1, \theta_2, \theta_3$, there exists a set of dual Euler angles $\theta_a, \theta_b, \theta_c$ such that*

$$R_i^{\alpha\alpha}(\theta_1) R_{i+1}^{\beta\beta}(\theta_2) R_i^{\alpha\alpha}(\theta_3) = R_{i+1}^{\beta\beta}(\theta_a) R_i^{\alpha\alpha}(\theta_b) R_{i+1}^{\beta\beta}(\theta_c).$$

In terms of a diagram this relation is given by:



The relation between both sets of Euler angles is given by (3.9) and (3.10).

LEMMA 3.8. *Let $\alpha, \beta \in \{x, y, z\}$, $\alpha \neq \beta$, $i \in \{1, \dots, N-1\}$. For every set of Euler angles $\theta_1, \theta_2, \theta_3$, there exists a set of dual Euler angles $\theta_a, \theta_b, \theta_c$ such that*

$$\begin{aligned} R_i^{\alpha\alpha}(\theta_1) R_i^{\beta}(\theta_2) R_i^{\alpha\alpha}(\theta_3) &= R_i^{\beta}(\theta_a) R_i^{\alpha\alpha}(\theta_b) R_i^{\beta}(\theta_c), \\ R_i^{\alpha\alpha}(\theta_1) R_{i+1}^{\beta}(\theta_2) R_i^{\alpha\alpha}(\theta_3) &= R_{i+1}^{\beta}(\theta_a) R_i^{\alpha\alpha}(\theta_b) R_{i+1}^{\beta}(\theta_c), \end{aligned}$$

or as a circuit diagram:



The relation between both sets of Euler angles is given by (3.9) and (3.10).

Lemmas 3.7 and 3.8 follow from the observation that the matrices involved have the same group structure as $SU(2)$. We refer the interested reader to [22] for further details.

4. Classical Ising model. In this section we show, based on the results from Section 3, that the approximate time-evolution operator from (2.3) can be implemented in a circuit of depth $\mathcal{O}(1)$ for Hamiltonians that are known as (classical) Ising models. The Ising model is a classical Hamiltonian because all the terms in its expansion commute. This allows for the $\mathcal{O}(1)$ depth. The Hamiltonians in Sections 6 and 7 consist of terms that do not commute and are truly quantum. They will require deeper circuits that are more challenging to compute.

The Hamiltonian for the Ising model is given by,

$$(4.1) \quad H(t) = \sum_{i=1}^{N-1} J_i^{\alpha}(t) \sigma_i^{\alpha} \sigma_{i+1}^{\alpha} + \sum_{i=1}^N h_i^{\alpha}(t) \sigma_i^{\alpha}, \quad \alpha \in \{x, y, z\}.$$

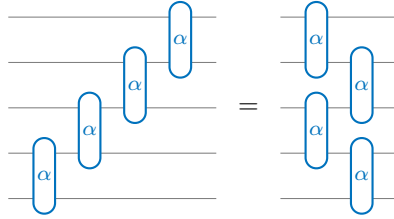
We will use the shorthand notation $H^{\alpha\alpha+\alpha}(t)$ for this Hamiltonian thereby referring to its nonzero terms. If we decompose this Hamiltonian in its two-spin and single-spin interaction, $H^{\alpha\alpha+\alpha}(t) = H^{\alpha\alpha}(t) + H^{\alpha}(t)$, a single time-step in the discretized

time-evolution operator (2.3) becomes,

$$\begin{aligned} U_\tau(\Delta t) &= \exp(-iH_\tau^{\alpha\alpha}\Delta t) \exp(-iH_\tau^\alpha\Delta t), \\ &= \exp\left(-i\sum_{i=1}^{N-1} J_i^\alpha(t_\tau) \sigma_i^\alpha \sigma_{i+1}^\alpha \Delta t\right) \exp\left(-i\sum_{i=1}^N h_i^\alpha(t_\tau) \sigma_i^\alpha \Delta t\right), \\ &= \prod_{i=1}^{N-1} R_i^{\alpha\alpha}(2J_i^\alpha(t_\tau)\Delta t) \prod_{i=1}^N R_i^\alpha(2h_i^\alpha(t_\tau)\Delta t). \end{aligned}$$

Since all the terms in $H^{\alpha\alpha+\alpha}(t)$ commute according to (3.1) and (3.5), we did not introduce a Trotter error according to (2.2), except for the discretization in time. Using the commutativity of two-spin Pauli rotations, Lemma 3.2(i), it follows that we can rearrange the ascending cascades of two-spin Pauli rotations that make up a single time-step into an even-odd ordering:

$$\prod_{i=1}^{N-1} R_i^{\alpha\alpha}(2J_i^\alpha(t_\tau)\Delta t) = \prod_{\text{odd } i} R_i^{\alpha\alpha}(2J_i^\alpha(t_\tau)\Delta t) \prod_{\text{even } i} R_i^{\alpha\alpha}(2J_i^\alpha(t_\tau)\Delta t)$$



The time-evolution operator (2.3) for the Ising becomes a horizontal concatenation of these layers interleaved with layers of single-spin rotations. According to Lemma 3.2 all gates in the circuit commute with each other and according to Lemma 3.3, gates of the same type that are acting on the same spin(s) can be fused together. This circuit compression method for Ising models is illustrated in Figure 1.

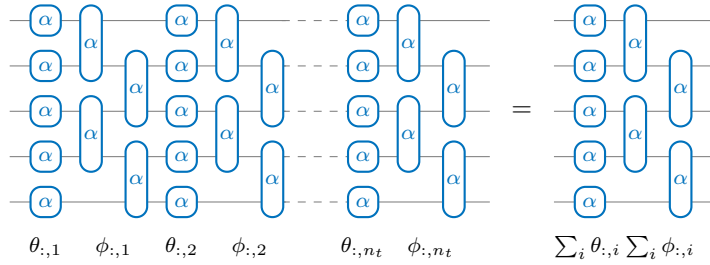
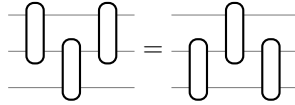


FIG. 1. Compression to a circuit of depth $\mathcal{O}(1)$ for time-evolution of an Ising model Hamiltonian as in (4.1).

The parameters of the compressed $\mathcal{O}(1)$ circuits are computed by straightforward numerical integration of the parameters of the time-dependent Ising Hamiltonians. This requires $\mathcal{O}(n_t N)$ operations for disordered Ising Hamiltonians and only $\mathcal{O}(n_t)$ for ordered Ising Hamiltonians. In case the Ising Hamiltonian is time-independent, the complexity can be further reduced to $\mathcal{O}(\log_2(n_t)N)$ and $\mathcal{O}(\log_2(n_t))$ for respectively disordered and ordered Ising Hamiltonians by applying the merging algorithm recursively.

5. Constant-depth circuits with fusion and turnover operations. In this section we present a second algorithm for computing constant-depth circuits for time-evolution by means of compressing longer circuits. The difference with the circuits presented in Section 4 is the type of operations that we can perform on the gates in the circuit. We constructively show that quantum circuits comprised of non-commuting two-spin gates which allow for a fusion and turnover operation can efficiently be compressed to a circuit with depth $\mathcal{O}(N)$. To show this, we use circuit transformations that are equivalent to transformations used in core chasing algorithms for eigenvalue problems [3, 39, 40].

The turnover operation acts locally on a pattern of three two-spin gates and changes a \vee -shaped pattern to a \wedge -shaped pattern or vice versa:



Lemma 3.7 was a first example of a turnover operation for two-spin Pauli rotations. A useful operation that we can do with the turnover operation is illustrated in Figures 2a to 2e. It pulls a *free gate* through an *ascending cascade* of gates which moves the incoming gate on position down.

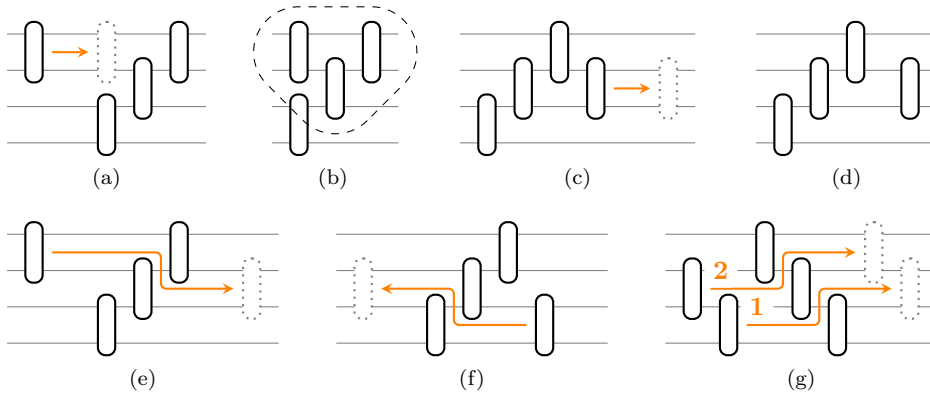


FIG. 2. (a) A free gate is moved in from the left side of the ascending cascade of three gates, (b) this gives a \vee -shaped pattern of gates that are ready for a turnover, (c) the turnover operation results in a free gate on the right side of the ascending cascade that has moved one position down, to achieve final configuration shown in (d). Steps (a)-(d) are summarized with the concise notation in (e). (f) A free gate can be moved from right to left, and (g) the order of turnover operations matters when moving multiple gates through a cascade.

An analogous operation can be performed for a free gate and a *descending cascade* of gates, or for a free gate on the right side of an ascending (Figure 2f) or descending cascade of gates. If we move multiple gates through a descending or ascending cascade, the order of turnover operations matters as illustrated in Figure 2g.

5.1. Square and triangle circuits. We define two types of circuits with fixed patterns of two-spin gates: a circuit with a *square* pattern of gates and with a *triangle* pattern of gates.

DEFINITION 5.1 (Square Circuit). A square circuit on N spins has N vertical layers that are alternately starting from the first and second spin.

DEFINITION 5.2 (Triangle Circuit). *A triangle circuit on N spins has $N - 1$ descending cascades with the i th descending cascade starting from spin $N - i$ and containing i two-spin gates.*

Figure 3 illustrates square and triangular circuits for the odd and even number of spins. The number of gates in a square circuit is equal to the number of gates in a triangle pattern and scales quadratically as $N(N - 1)/2$.

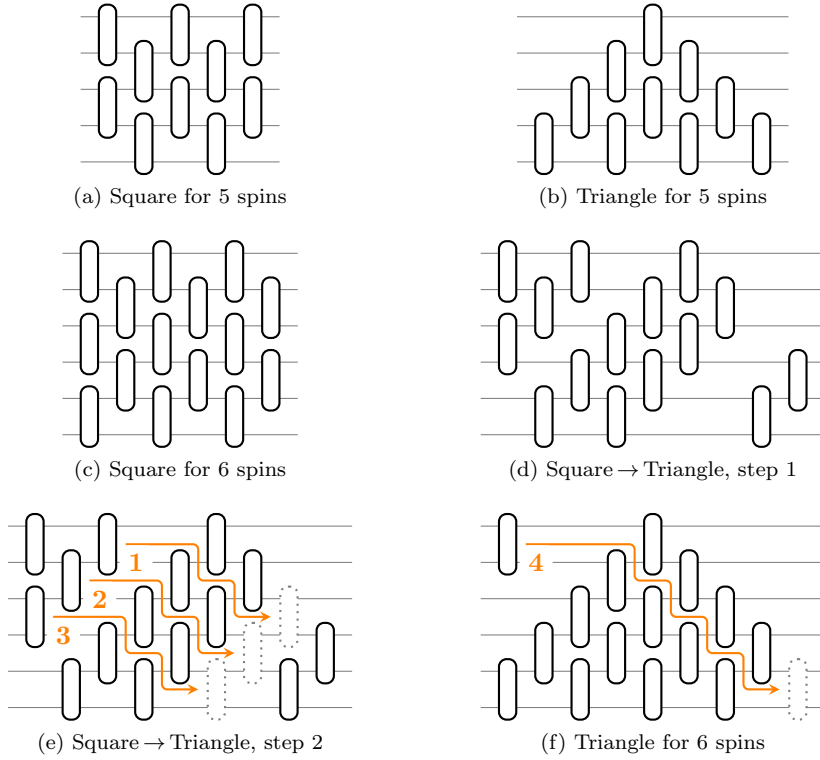


FIG. 3. (a) Square and (b) triangle circuits for systems with 5 spins. (c) A square circuit on 6 spins and (d) the same square circuit with additional open space. (e) The three gates in the second ascending cascade of the square circuit are moved over to the bottom half using a total of 6 turnover operations, (f) finally the first gate is moved from the top left to bottom right using 4 turnover operations resulting in a triangle circuit for 6 spins.

Using the turnover operation, we can transform a square circuit into a triangle circuit. The algorithm for 6 spins is summarized in Figure 3 and generalizes to any even number of spins. In Figure 3d we create sufficient open space in the square circuit such that we can use the turnover operation to move the top half of gates in the square circuit over to the bottom half in Figure 3e, thereby creating the triangle circuit in Figure 3f. With minor alterations the algorithm to convert a square to a triangle circuit can be extended to systems with an odd number of spins. Furthermore, all the turnover operations are reversible, which means that we can easily transform a triangle circuit back to a square circuit by reversing the algorithm. The computational complexity to go from a square to a triangle circuit on N spins or vice versa scales as $\mathcal{O}(N^3)$.

5.2. Merging gates with a triangle circuit. We will now proceed to show that a triangle is the minimal circuit for two-spin gates that allow for a turnover and

fusion operation. This holds because every two-spin gate on either the left or right side of a triangle circuit can be repeatedly turned over with gates in the triangle circuit until it eventually can be fused with a gate at the bottom of the triangle circuit. This is illustrated in [Figures 4a](#) and [4b](#) for a 5-spin triangle circuit with a cascade of gates on both the left and right side.

For a Trotter-based implementation of a simulation circuit, the gates often come in vertical layers that act alternating on even and odd spins. In that case the annihilation of the gates acting on even spins can all be done in parallel, and the same is true for the gates acting on odd spins. This is shown in [Figures 4c](#) and [4d](#).

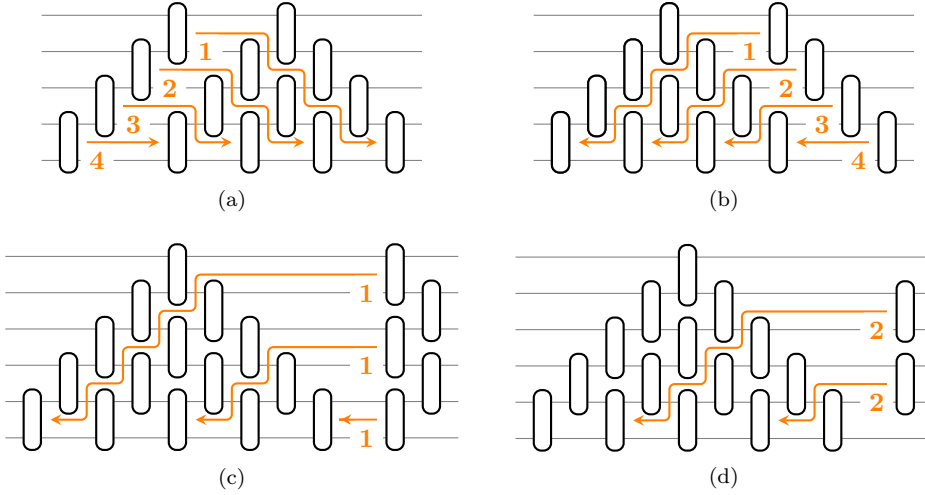


FIG. 4. (a) Sequential annihilation of an ascending sequence of gates positioned to the left of a 5-spin triangle circuit through repeated turnover and fuse operations, (b) similar for a sequence of gates on the right of a 5-spin triangle circuit. A vertical layer of gates can be merged in parallel with a 6-spin triangle circuit by first merging all gates acting on odd spins (c) and afterwards merging the remaining gates on the even spins (d).

The average computational cost of merging a single gate with a triangle circuit scales as $\mathcal{O}(N)$, and the cost of merging a complete vertical layer of gates with a triangle is $\mathcal{O}(N^2)$. There is no difference in computational complexity between disordered and ordered Hamiltonians, both require $\mathcal{O}(n_t N^2)$ operations to annihilate n_t time-step. For time-independent Hamiltonians, we can again reduce the complexity to $\mathcal{O}(\log_2(n_t) N^2)$ by using the merging algorithm recursively.

5.3. Circuit compression algorithms. The complete circuit compression algorithms are outlined in [Algorithm 5.1](#) for time-dependent Hamiltonians and in [Algorithm 5.2](#) for the time-independent case. The input to [Algorithm 5.1](#) is a Trotter circuit C on N spins with n_t time-steps or $2n_t$ vertical layers. We assume that $n_t > N/2$ as otherwise minimal depth is not reached. In line 1, we take the first N vertical layers out of C and transform it to triangle representation C' . The for loop runs over the remaining layers and merges them into C' using the approach of [Figures 4c](#) and [4d](#). In the end C' is transformed back to a square circuit of depth N that is equivalent to the input circuit C with depth $2n_t$.

In the time-independent case, we first create a minimal depth square C' from the fixed time-step C_{TS} in line 2. This square is again transformed to a triangle that is repeatedly merged with itself in line 6 by splitting the triangle into cascades and using

Algorithm 5.1 Compression of time-dependent Hamiltonians

Data: Trotter circuit C on N spins with n_t time-steps, $n_t > N/2$.**Result:** Compressed $N \times N$ Trotter circuit C' equivalent to C .

```

1  $C' \leftarrow \text{TriangleCircuit}(C[:, 1:N])$ 
2 for  $l \leftarrow N + 1$  to  $2n_t$  do
3   |  $\text{MergeLayer}(C', C[:, l])$ 
4 end
5  $C' \leftarrow \text{SquareCircuit}(C')$ 

```

the method from Figure 4b. This doubles the total simulation time in every loop.

Both algorithms end with a conversion to square format as this form has a shallower circuit depth compared to the triangle format. This makes square circuits better suited for NISQ devices, even though both circuits have the same number of gates.

Algorithm 5.2 Compression of time-independent Hamiltonians

Data: Trotter circuit C_{TS} for a single time-step, total number of time-steps $n_t = 2^\tau N/2$.**Result:** $N \times N$ Trotter circuit C' for n_t time-steps.

```

// Fill  $C'$  to square form with  $N/2$  time-steps  $C_{TS}$ .
1 for  $l \leftarrow 1$  to  $N/2$  do
2   |  $C'[:, l:l+1] \leftarrow C_{TS}$ 
3 end
4  $C' \leftarrow \text{TriangleCircuit}(C')$ 
// Repeatedly merge  $C'$  with itself up to total time-steps of  $2^\tau N/2$ .
5 for  $t \leftarrow 1$  to  $\tau$  do
6   |  $\text{MergeTriangle}(C', C')$ 
7 end
8  $C' \leftarrow \text{SquareCircuit}(C')$ 

```

6. Kitaev chains and XY models. In this section we show that two classes of closely related Hamiltonians, respectively known as Kitaev chains and XY models, can be implemented with gates that satisfy the fusion and turnover conditions introduced in the previous section. It follows that the discretized time-evolution operator (2.3) can be implemented in a quantum circuit with depth $\mathcal{O}(N)$ for these models.

6.1. Kitaev chains. A Kitaev chain is a Hamiltonian of the form

$$(6.1) \quad H(t) = \sum_{i=1}^{N-1} J_i^{\alpha_i}(t) \sigma_i^{\alpha_i} \sigma_{i+1}^{\alpha_i},$$

with the restriction that two neighboring spins i and $i + 1$ have different type of interaction, i.e. $\alpha_i \neq \alpha_{i+1}$. For example,

$$(6.2) \quad H(t) = J_1^y(t) \sigma_1^y \sigma_2^y + J_2^x(t) \sigma_2^x \sigma_3^x + J_3^z(t) \sigma_3^z \sigma_4^z + J_4^x(t) \sigma_4^x \sigma_5^x,$$

is a 5-spin Kitaev chain. Using a Trotter decomposition in even and odd terms, we get the following approximation for a single time-step:

$$\begin{aligned} U_\tau(\Delta t) &= \exp(-iH(t)\Delta(t)), \\ &= \exp(-iH_{\text{even}}(t)\Delta(t)) \exp(-iH_{\text{odd}}(t)\Delta(t)) + \mathcal{O}(\Delta t^2), \\ &\approx R_2^{xx}(2J_2^x(t_\tau)\Delta t) R_4^{xx}(2J_4^x(t_\tau)\Delta t) R_1^{yy}(2J_1^y(t_\tau)\Delta t) R_3^{zz}(2J_3^z(t_\tau)\Delta t). \end{aligned}$$

This is illustrated in a circuit diagram on the left side of [Figure 5](#). A square circuit for the Hamiltonian [\(6.2\)](#) that satisfies [Definition 5.1](#) is shown in the middle of [Figure 5](#). It consists of two complete time-steps and a single vertical layer of gates acting on the odd spins. Using the turnover operation from [Lemma 3.7](#), we can use the algorithm from [Section 5.1](#) to transform this square circuit to the equivalent triangle circuit shown on the right of [Figure 5](#). Remark that the number of gates of each type is changed due to [Lemma 3.7](#), but the type of gate acting on every pair of spins is preserved. Afterwards, we can use the turnover and fusion operation from [Lemma 3.3\(ii\)](#) to compress the circuit to constant depth by means of [Algorithm 5.1](#) or [Algorithm 5.2](#). This shows that we can always compress the circuit for the simulation of a Kitaev to square or triangle form.

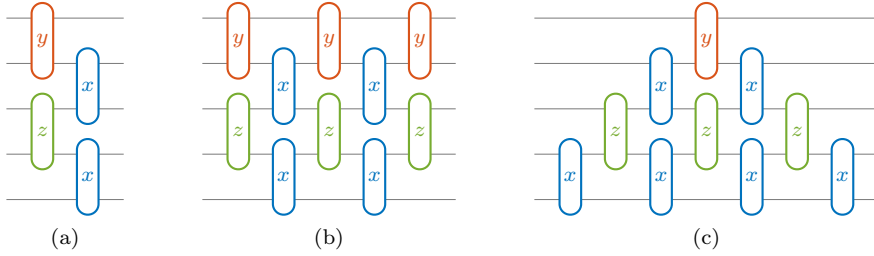


FIG. 5. (a) A single time-step for the Kitaev chain [\(6.2\)](#), (b) a square circuit for this Kitaev chain, and (c) a triangle circuit for this Kitaev chain.

6.2. XY models. The class of XY Hamiltonians is given by

$$(6.3) \quad H(t) = \sum_{i=1}^{N-1} J_i^\alpha(t) \sigma_i^\alpha \sigma_{i+1}^\alpha + J_i^\beta(t) \sigma_i^\beta \sigma_{i+1}^\beta, \quad \alpha, \beta \in \{x, y, z\},$$

where $\alpha \neq \beta$ and all parameters J_i^α, J_i^β are nonzero. This class consists of XY, XZ, and YZ Hamiltonians for the appropriate choices of α and β . If we split [\(6.3\)](#) in $H(t) = H_{\alpha\beta}(t) + H_{\beta\alpha}(t)$ with

$$\begin{aligned} H_{\alpha\beta}(t) &= \sum_{\text{odd } i} J_i^\alpha(t) \sigma_i^\alpha \sigma_{i+1}^\alpha + \sum_{\text{even } i} J_i^\beta(t) \sigma_i^\beta \sigma_{i+1}^\beta, \\ H_{\beta\alpha}(t) &= \sum_{\text{odd } i} J_i^\beta(t) \sigma_i^\beta \sigma_{i+1}^\beta + \sum_{\text{even } i} J_i^\alpha(t) \sigma_i^\alpha \sigma_{i+1}^\alpha, \end{aligned}$$

we see that we have rewritten it as a sum of two regular Kitaev chains. Using the Trotter decomposition and circuit compression described in [Section 6.1](#) for each of the two Kitaev chains $H_{\alpha\beta}(t)$ and $H_{\beta\alpha}(t)$, we find that we can get a circuit for $H(t)$ as a product of two triangle circuits for the Kitaev chain. This is illustrated in the

first row of Figure 6 for the case of the XY model. It follows from the commutativity properties (i) and (ii) of Lemma 3.2 that we can combine these two Kitaev chains into one triangle circuit, as shown on the second row of Figure 6. The gates in this circuit are two-axes rotation gates defined in Lemma 3.4, and are a product of R^{xx} and R^{yy} rotations. Because of the commutativity of the two Kitaev chains, they can be simulated separately. However, for certain types of quantum hardware, a product of R^{xx} and R^{yy} rotations can be evaluated at approximately the same cost as a single two-spin Pauli rotation [7]. In that case it makes sense to simulate the two Kitaev chains simultaneously.

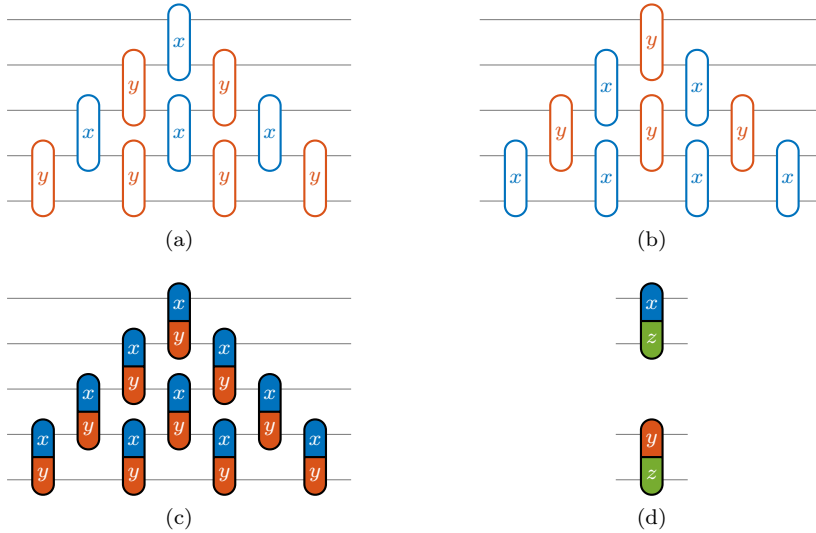


FIG. 6. Triangle circuits for the Kitaev chains H_{XY} (a) and H_{YX} (b). (c) Combination of two Kitaev chains in a single triangle circuit for the complete XY model, (d) two-spin gates for XZ and YZ models.

The elementary gates for XY, XZ and YZ Hamiltonians are listed in Figures 6c and 6d. Within each class, these gates can be fused and turned over by separating them into their two-spin Pauli rotations and using the results from Section 3.

7. Transverse-field XY and Ising models. In this section, we discuss the most general class of Hamiltonians for which our compression algorithm works. These are the TFXY Hamiltonians already introduced in (2.5). In Section 7.1 we briefly discuss the case of the full TFXY model, but more details about the implementation of the fusion and turnover operation are deferred to Section 8. Section 7.2 introduces the transverse-field Ising model (TFIM) as a special case of TFXY that can be treated separately under some conditions.

7.1. TFXY model. The two-spin gates for TFXY, TFXZ, and TFYZ Hamiltonians are shown in Figure 7. The existence of the TFXY fusion and turnover operations is proven in [22] by considering the Hamiltonian algebras of 2- and 3-spin TFXY models and their Cartan decompositions. It is further shown that the TFXY fusion requires two turnovers of Euler decompositions of $SU(2)$, Lemma 3.6, and that a TFXY turnover can be done by invoking this lemma 32 times. In our compression algorithms we use more efficient implementations of the fusion and turnover operations described in Section 8. With these two operations, we can use Algorithms 5.1

and 5.2 to compress TFXZ circuits.

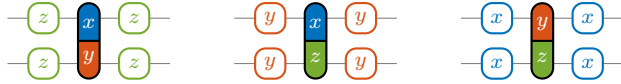


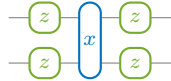
FIG. 7. Two-spin gates for TFXZ, TFXZ, and TFYZ Hamiltonians.

7.2. TFIM model. The transverse-field Ising model is a special case of the TFXZ Hamiltonian with only one non-zero coupling term,

$$(7.1) \quad H(t) = \sum_{i=1}^{N-1} J_i^\alpha(t) \sigma_i^\alpha \sigma_{i+1}^\alpha + \sum_{i=1}^N h_i^\beta(t) \sigma_i^\beta, \quad \alpha, \beta \in \{x, y, z\},$$

where $\alpha \neq \beta$. The most frequently studied cases in the literature are $\alpha = x, \beta = z$, and $\alpha = z, \beta = x$. We present our discussion for the former case, but all results remain valid for other choices.

One straightforward approach for compressing a TFIM circuit is to use the more general TFXZ gates from Section 7.1 and set the J^y parameters to zero. While this leads to a valid compression algorithm, the J^y parameters in the triangle circuit will become nonzero throughout the procedure as the gate



doesn't allow for fusion and turnover operations.

An alternative approach is to use Lemma 3.8 as a turnover operation for a TFIM Hamiltonian. In this setting, the turnover simultaneously operates on one- and two-spin gates and the compression algorithm appears to be different from the algorithms in Section 5, but it turns out that they are completely analogous. We illustrate the gist of the idea for a small example that draws a parallel between a 6-spin Kitaev chain and a 3-spin TFIM circuit in Figure 8. We see that we can map the two-spin R^{zz} rotations from the Kitaev chain to the one-spin R^z rotations in the TFIM Hamiltonian and the R^{xx} rotations are mapped to R^{xx} rotations that mutually commute (Lemma 3.2). Another interpretation is that every pair of consecutive spins in the Kitaev chain is combined to a single spin in the TFIM circuit. It follows from this mapping that we can use the algorithms from Section 5 together with Lemmas 3.3 and 3.8 to compress TFIM circuits and transform them from square to triangle or vice versa. The total number of gates for a minimal representation of an N -spin TFIM circuit is $N(2N-1)$, N^2 of the gates are one-spin Pauli rotations, $N(N-1)$ are two-spin Pauli rotations.

Depending on the details of the quantum hardware, it is important to remark that the cost of a single two-spin Pauli-X rotation can sometimes be considered approximately the same as the cost of a two-spin XY rotation since both require the same number of two qubit CNOT gates [7]. These are the main source of errors on many devices and in that case we expect the TFXZ mapping to perform better. However, for other devices the $R^{\alpha\alpha}$ transformations can be supported natively and the TFIM mapping might be preferable.

8. Implementation details. We describe some relevant details of the numerical implementations of our fusion and turnover algorithms.

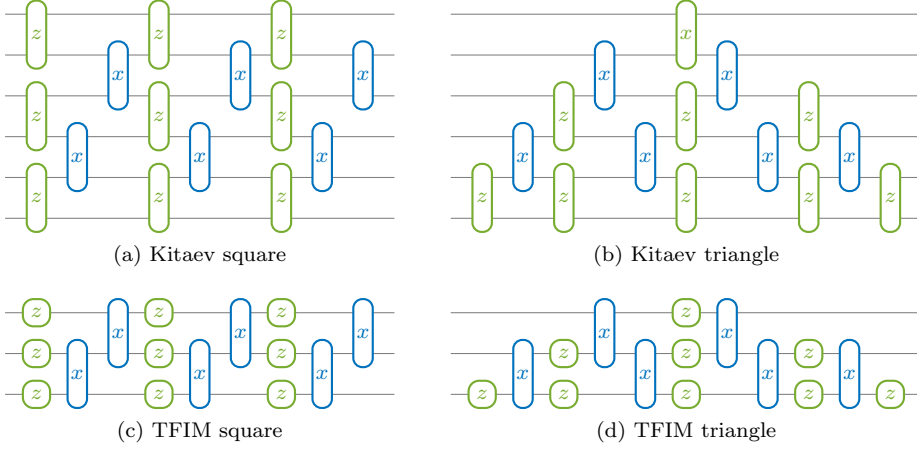


FIG. 8. (a) A square circuit for a Kitaev chain with $2N = 6$ spins and (b) the corresponding triangle circuit. These can be mapped to a (c) square circuit for a TFIM Hamiltonian for $N = 3$ spins and (d) a triangle circuit for the TFIM Hamiltonian.

8.1. Storage. All Pauli rotations throughout the compression algorithms for Ising, Kitaev chains, XY, and TFIM models are stored by two doubles storing $\cos(\theta/2)$ and $\sin(\theta/2)$. This avoids the numerical evaluation of (inverse) trigonometric functions, and instead we can rely on Givens rotation matrices in the turnover operations.

For the fusion of two compatible 1- or 2-spin Pauli rotations (Lemma 3.3), we simply compute the first column of a product of two 2×2 matrices.

8.2. Turnover of $SU(2)$. The angles for an $SU(2)$ turnover are given by (3.9) and (3.10). Numerical evaluation of these formulas requires (inverse) trigonometric functions which can be avoided and replaced by Givens rotations leading to ideal numerical roundoff properties [3].

As input to our $SU(2)$ turnover routine, we have three Euler angles stored as

$$(8.1) \quad [\cos(\theta_a/2) \quad \sin(\theta_a/2)], \quad [\cos(\theta_b/2) \quad \sin(\theta_b/2)], \quad [\cos(\theta_c/2) \quad \sin(\theta_c/2)],$$

or, in short, $[c_a, s_a]$, $[c_b, s_b]$, and $[c_c, s_c]$. Without loss of generality, we form the first column of the corresponding $SU(2)$ matrix (3.6) under the assumption that the Euler angles correspond to a YZY parametrization (3.11), $R^y(\theta_1)R^z(\theta_2)R^y(\theta_3)$. The (1, 1) element, $\alpha = \alpha_r + i\alpha_i$, and the (2, 1) element, $\beta = \beta_r + i\beta_i$, of the $SU(2)$ matrix are in that case:

$$(8.2) \quad \begin{aligned} \alpha_r &= c_b(c_a c_c - s_a s_c), & \alpha_i &= -s_b(c_a c_c + s_a s_c), \\ \beta_r &= c_b(s_a c_c + c_a s_c), & \beta_i &= -s_b(s_a c_c - c_a s_c), \end{aligned}$$

which we directly compute from our input Euler angles (8.1). The dual Euler angles that we have to compute form a ZYZ decomposition (3.8) of the same $SU(2)$ matrix, so they have to satisfy:

$$(8.3) \quad \begin{aligned} \alpha_r &= c_2(c_1 c_3 - s_1 s_3), & \alpha_i &= -c_2(s_1 c_3 + c_1 s_3), \\ \beta_r &= s_2(c_1 c_3 + s_1 s_3), & \beta_i &= s_2(s_1 c_3 - c_1 s_3). \end{aligned}$$

It follows from (3.8) and (8.3) that we can compute $c_2 = |\alpha| = \sqrt{\alpha_r^2 + \alpha_i^2}$ and $s_2 = |\beta| = \sqrt{\beta_r^2 + \beta_i^2}$, which is properly normalized as $|\alpha|^2 + |\beta|^2 = 1$ by (3.6).

Next, we get from (8.3) that we can compute $[c_1, s_1]$ and $[c_3, s_3]$ from two Givens rotation matrices that introduce zeros in, respectively, v_1 or v'_1 , and v_3 or v'_3 ,

$$(8.4) \quad \begin{aligned} v_1 &= \begin{bmatrix} \alpha_r/c_2 + \beta_r/s_2 \\ -\alpha_i/c_2 + \beta_i/s_2 \end{bmatrix}, & v'_1 &= \begin{bmatrix} -\alpha_i/c_2 - \beta_i/s_2 \\ \beta_r/s_2 - \alpha_r/c_2 \end{bmatrix}, \\ v_3 &= \begin{bmatrix} -\alpha_i/c_2 + \beta_i/s_2 \\ -\alpha_r/c_2 + \beta_r/s_2 \end{bmatrix}, & v'_3 &= \begin{bmatrix} \alpha_r/c_2 + \beta_r/s_2 \\ -\alpha_i/c_2 - \beta_i/s_2 \end{bmatrix}. \end{aligned}$$

To ensure the lowest relative error, we choose the vector with largest norm between v_1 and v'_1 , and similar for v_3 and v'_3 . It follows from the roundoff properties of Givens rotation matrices [3] that this algorithm computes the dual Euler angles $[c_1, s_1]$, $[c_2, s_2]$ and $[c_3, s_3]$ to high relative accuracy.

8.3. TFX Y gates. A TFX Y gate is parametrized by six angles and has the following matrix representation,

$$(8.5) \quad U_{\text{TFXY}} = \begin{array}{ccc} \theta_1 & \theta_3 & \theta_5 \\ \textcircled{z} & \textcircled{x} & \textcircled{z} \\ \theta_2 & \theta_4 & \theta_6 \\ \textcircled{z} & \textcircled{y} & \textcircled{z} \end{array} = \begin{bmatrix} \alpha & & & -\bar{\delta} \\ & \beta & -\bar{\gamma} & \\ & \gamma & \bar{\beta} & \\ \delta & & & \bar{\alpha} \end{bmatrix},$$

where the matrix entries are given by

$$(8.6) \quad \begin{aligned} \alpha &= \cos((\theta_3 - \theta_4)/2) e^{-i(\theta_1 + \theta_2 + \theta_5 + \theta_6)/2}, \\ \beta &= \cos((\theta_3 + \theta_4)/2) e^{-i(\theta_1 - \theta_2 + \theta_5 - \theta_6)/2}, \\ \gamma &= -i \sin((\theta_3 + \theta_4)/2) e^{-i(\theta_1 - \theta_2 - \theta_5 + \theta_6)/2}, \\ \delta &= -i \sin((\theta_3 - \theta_4)/2) e^{-i(\theta_1 + \theta_2 - \theta_5 - \theta_6)/2}. \end{aligned}$$

It can be shown that both the outer 2×2 matrix $\begin{bmatrix} \alpha & -\bar{\delta} \\ \delta & \bar{\alpha} \end{bmatrix}$ and the inner 2×2 matrix $\begin{bmatrix} \beta & -\bar{\gamma} \\ \gamma & \bar{\beta} \end{bmatrix}$ form $SU(2)$ by mapping the matrix elements to two independent ZYZ decompositions (3.8). In our implementation, we store a TFX Y gate by the four complex numbers α , β , γ , and δ which can be easily computed from the angles via (8.6). The values α , β , γ , and δ can easily be converted back to a parametrization with six angles.

Unitary 4×4 matrices that have the nonzero pattern of the matrix in (8.5) are also known as *matchgates* [7]. Matchgates can be permuted to a block-diagonal matrix by the following permutation matrix:

$$(8.7) \quad P_{\oplus} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}, \quad P_{\oplus}^T = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}.$$

We get that,

$$(8.8) \quad P_{\oplus} \begin{bmatrix} \alpha & & & -\bar{\delta} \\ & \beta & -\bar{\gamma} & \\ & \gamma & \bar{\beta} & \\ \delta & & & \bar{\alpha} \end{bmatrix} P_{\oplus}^T = \begin{bmatrix} \alpha & -\bar{\delta} & & \\ \delta & \bar{\alpha} & & \\ & & \beta & -\bar{\gamma} \\ & & \gamma & \bar{\beta} \end{bmatrix} = \begin{bmatrix} \alpha & -\bar{\delta} \\ \delta & \bar{\alpha} \end{bmatrix} \oplus \begin{bmatrix} \beta & -\bar{\gamma} \\ \gamma & \bar{\beta} \end{bmatrix}.$$

In what follows, we use the notation $\text{TFXY}(A, B)$, where $A, B \in SU(2)$, to denote the TFX Y gate with its outer 2×2 $SU(2)$ matrix equal to A and its middle 2×2 $SU(2)$

This unitary matrix has a lot of structure that we exploit. The four 2×2 matrix pairs (Q_{11}, Q_{33}) , (Q_{22}, Q_{44}) , (Q_{23}, Q_{41}) , (Q_{14}, Q_{32}) are all of the form:

$$(8.12) \quad \left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}, \begin{bmatrix} \bar{d} & -\bar{c} \\ -\bar{b} & \bar{a} \end{bmatrix} \right).$$

This means that we only have to compute four 2×2 matrices to construct Q . Furthermore, it can be shown that matrix pairs of this form can always be simultaneously (anti-)diagonalized. A second property that we use, is that the four diagonal blocks are of equal norm, and similarly the four blocks on the anti-diagonal are of equal norm. To compute the turnover, we have to compute the following four coupled matrix decompositions:

$$\begin{aligned} U^*(Q_{11}, Q_{33})Y^* &= ([{}^{w_{11}} \ x_{11}], [{}^{x_{22}} \ w_{22}]), & V^*(Q_{22}, Q_{44})Z^* &= ([{}^{w_{11}} \ x_{11}], [{}^{x_{22}} \ w_{22}]), \\ U^*(Q_{14}, Q_{32})Z^* &= ([{}_{x_{12}} \ w_{12}], [{}_{w_{21}} \ x_{21}]), & V^*(Q_{23}, Q_{41})Y^* &= ([{}_{x_{12}} \ w_{12}], [{}_{w_{21}} \ x_{21}]), \end{aligned}$$

in such a way that the ordering of the (anti-)diagonalizations is consistent and that they share the same eigenvectors.

Our numerical algorithm depends on the ratio between $\|Q_{i,i}\|_2$ and $\|Q_{k,5-k}\|_2$. If $\|Q_{i,i}\|_2 \leq \|Q_{k,5-k}\|_2$, we start with computing $SU(2)$ matrices U and Y that diagonalize (Q_{11}, Q_{33}) . Keeping Y fixed, we compute V that anti-diagonalizes $(Q_{23}, Q_{41})Y^*$. Finally keeping V fixed, we compute Z that diagonalizes $V^*(Q_{22}, Q_{44})$. If $\|Q_{i,i}\|_2 > \|Q_{k,5-k}\|_2$, we start with computing $SU(2)$ matrices V and Y that anti-diagonalize (Q_{23}, Q_{41}) . Keeping Y fixed, we compute U that diagonalizes $(Q_{11}, Q_{33})Y^*$. Finally keeping U fixed, we compute Z that anti-diagonalizes $U^*(Q_{14}, Q_{32})$.

Afterwards, the $SU(2)$ matrices W and X are determined from the (anti-)diagonal elements of the (anti-)diagonalized matrices. To compute the turnover in the other direction (\wedge to \vee), from the right-hand side of (8.9) to the left-hand side, we simply interchange the roles of the first and third spins. This is just a permutation of the 8×8 unitary. During our extensive numerical tests we observed that this algorithm always computes the TFXY turnover up to high relative accuracy.

9. Numerical examples. All numerical experiments are performed on a AMD Ryzen Threadripper 3990X 64-Core Processor @ 2.9 GHz with 256 GB RAM. Our experiments can be reproduced with the F3C++ code [12, 37].

For our first numerical experiment, we compressed Trotter circuits for XY and TFXY models with randomly generated angles sampled from the standard normal distribution to benchmark the speed and accuracy of our algorithms. Figure 9 shows the cubic scaling of transforming the circuit between square and triangle representation and back, and the quadratic scaling of merging a time-step with a triangle circuit. Both algorithms easily scale up to $\mathcal{O}(10^3)$ spins, which is well beyond the capabilities of current quantum hardware. The compression of a full TFXY model, based on the simultaneous diagonalization algorithm for the turnover operation, is about a factor of 10 slower than compressing an XY model, which uses two $SU(2)$ turnovers. The TFXY turnover can alternatively be implemented with 32 $SU(2)$ turnovers [22], hence the simultaneous diagonalization approach is 1.5 times faster. Figure 10 displays the Frobenius norm difference between the unitary of the full Trotter circuit Q_f , and the unitary of the compressed Trotter circuit Q_c in function of the number of time-steps for systems up to 10 spins. We observe that our compression algorithm achieves high accuracy in all cases and that the roundoff error grows sub-linearly.

For our final numerical experiment, we report the error on the compression algorithms for the adiabatic state preparation experiment presented in [22]. In this

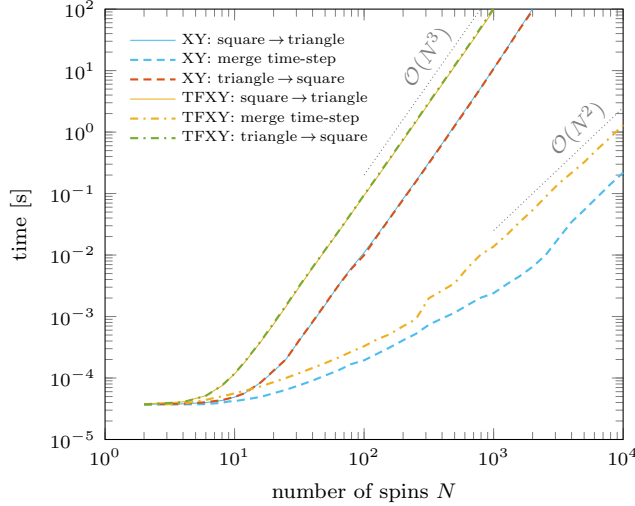


FIG. 9. Scaling for compression of XY and TFX models in function of number of spins.

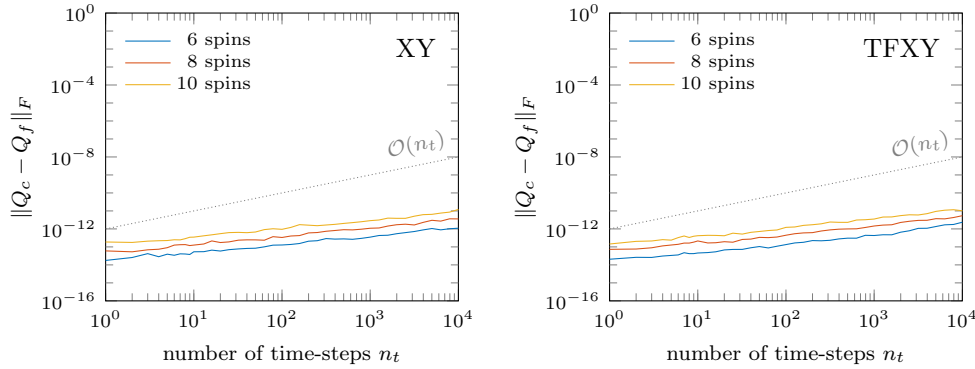


FIG. 10. Maximum Frobenius error on 100 randomly generated compressed circuits (Q_c) compared to full circuits for XY and TFX models in function of number of time-steps.

experiment, the ground state of a classical Ising model, that is easy to prepare, is evolved adiabatically to the ground state of a TFIM Hamiltonian. The experiment in [22] is performed for a 5-spin system with two different time-steps Δt , 0.05 and 0.25, and n_t respectively chosen as 1200 and 240 such that the final simulation time is the same. The first option leads to a significantly smaller Trotter error but to a slightly higher numerical error on the compressed circuit due to roundoff accumulation as five times more time-steps have to be merged. The total error remains small in all cases.

10. Conclusions. We presented a fast and accurate quantum circuit synthesis algorithm that is suitable for compressing circuits to simulate certain classes of spin Hamiltonians known as free fermionizable models on current generation quantum devices. Our algorithms are based on matrix factorizations and easily scale up to $\mathcal{O}(10^3)$ spins as we make use of localized circuit transformations that allow us to keep the $2^N \times 2^N$ unitary matrix in factorized form throughout the algorithm. Furthermore, we can make the Trotter error as small as required without increasing the circuit depth. Numerical experiments showed that our turnover routines based on Givens

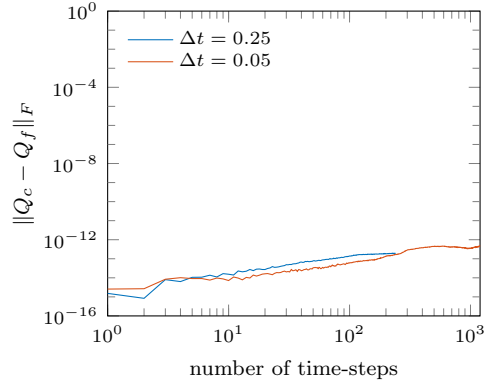


FIG. 11. Frobenius error on compressed circuit (Q_c) compared to full circuits (Q_f) for the adiabatic state preparation experiment presented in [22].

rotation matrices and simultaneous (anti-)diagonalization behave backward stable. Our approach is not limited to Hamiltonian simulation, any quantum circuit partially comprised of the gates that we discussed admits local fusion and turnover operations. We provide reference implementations that are readily available and can be used to improve existing quantum compilers and transpilers.

REFERENCES

- [1] F. ARUTE, K. ARYA, R. BABBUSH, D. BACON, J. C. BARDIN, ET AL., *Quantum supremacy using a programmable superconducting processor*, Nature, 574 (2019), pp. 505–510, <https://doi.org/10.1038/s41586-019-1666-5>.
- [2] F. ARUTE, K. ARYA, R. BABBUSH, D. BACON, J. C. BARDIN, ET AL., *Hartree-Fock on a superconducting qubit quantum computer*, Science, 369 (2020), <https://doi.org/10.1126/science.abb9811>.
- [3] J. L. AURENTZ, T. MACH, L. ROBOL, R. VANDEBRIL, AND D. S. WATKINS, *Core-Chasing Algorithms for the Eigenvalue Problem*, Fundamentals of Algorithms, SIAM, 2018.
- [4] A. BARENCO, C. H. BENNETT, R. CLEVE, D. P. DIVINCENZO, N. MARGOLUS, ET AL., *Elementary gates for quantum computation*, Phys. Rev. A, 52 (1995), pp. 3457–3467, <https://doi.org/10.1103/physreva.52.3457>.
- [5] R. BARENDTS, A. SHABANI, L. LAMATA, J. KELLY, A. MEZZACAPO, ET AL., *Digitized adiabatic quantum computing with a superconducting circuit*, Nature, 534 (2016), pp. 222–226, <https://doi.org/10.1038/nature17658>.
- [6] L. BASSMAN, M. URBANEK, M. METCALF, J. CARTER, A. F. KEMPER, ET AL., *Simulating quantum materials with digital quantum computers*, 2021, <https://arxiv.org/abs/2101.08836>.
- [7] L. BASSMAN, R. VAN BEEUMEN, E. YOUNIS, E. SMITH, C. IANCU, ET AL., *Constant-depth circuits for dynamic simulations of materials on quantum computers*, 2021, <https://arxiv.org/abs/2103.07429>.
- [8] B. BAUER, S. BRAVYI, M. MOTTA, AND G. K.-L. CHAN, *Quantum algorithms for quantum chemistry and quantum materials science*, Chem. Rev., 120 (2020), pp. 12685–12717, <https://doi.org/10.1021/acs.chemrev.9b00829>.
- [9] D. W. BERRY, A. M. CHILDS, R. CLEVE, R. KOTHARI, AND R. D. SOMMA, *Simulating Hamiltonian dynamics with a truncated Taylor series*, Phys. Rev. Lett., 114 (2015), p. 090502, <https://doi.org/10.1103/PhysRevLett.114.090502>.
- [10] D. W. BERRY, A. M. CHILDS, AND R. KOTHARI, *Hamiltonian simulation with nearly optimal dependence on all parameters*, FOCS, 2015-Decem (2015), pp. 792–809, <https://doi.org/10.1109/FOCS.2015.54>.
- [11] D. CAMPS AND R. VAN BEEUMEN, *Approximate quantum circuit synthesis using block encodings*, Phys. Rev. A, 102 (2020), p. 52411, <https://doi.org/10.1103/PhysRevA.102.052411>.
- [12] D. CAMPS AND R. VAN BEEUMEN, *F3C*, 2021, <https://doi.org/10.5281/zenodo.5160761>, <https://github.com/QuantumComputingLab/f3c> (accessed 2021/08/04). Version 0.1.0.

- [13] D. CAMPS AND R. VAN BEEUMEN, *QCLAB*, 2021, <https://doi.org/10.5281/zenodo.5160555>, <https://github.com/QuantumComputingLab/qclab> (accessed 2021/08/04). Version 0.1.2.
- [14] A. M. CHILDS, Y. SU, M. C. TRAN, N. WIEBE, AND S. ZHU, *Theory of Trotter error with commutator scaling*, Phys. Rev. X, 11 (2021), <https://doi.org/10.1103/PhysRevX.11.011020>.
- [15] A. GILYÉN, Y. SU, G. H. LOW, AND N. WIEBE, *Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics*, 2018, <https://arxiv.org/abs/1806.01838>.
- [16] J. HAAH, M. B. HASTINGS, R. KOTHARI, AND G. H. LOW, *Quantum algorithm for simulating real time evolution of lattice Hamiltonians*, SIAM J. Comput., (2021), pp. 250–284, <https://doi.org/10.1137/18m1231511>.
- [17] N. HATANO AND M. SUZUKI, *Finding exponential product formulas of higher orders*, in Quantum Annealing and Other Optimization Methods, Springer Berlin Heidelberg, 2005, pp. 37–68, https://doi.org/10.1007/11526216_2.
- [18] P. A. IVANOV, E. S. KYOSEVA, AND N. V. VITANOV, *Engineering of arbitrary $U(N)$ transformations by quantum Householder reflections*, Phys. Rev. A, 74 (2006), pp. 1–8, <https://doi.org/10.1103/PhysRevA.74.022323>.
- [19] Z. JIANG, K. J. SUNG, K. KECHEDZHI, V. N. SMELYANSKIY, AND S. BOIXO, *Quantum Algorithms to Simulate Many-Body Physics of Correlated Fermions*, Phys. Rev. Appl., 9 (2018), p. 044036, <https://doi.org/10.1103/PhysRevApplied.9.044036>.
- [20] A. KALEV AND I. HEN, *Simulating Hamiltonian dynamics with an off-diagonal series expansion*, 2020, <https://arxiv.org/abs/2006.02539>.
- [21] I. D. KIVLICHAN, J. MCCLEAN, N. WIEBE, C. GIDNEY, A. ASPURU-GUZIK, ET AL., *Quantum Simulation of Electronic Structure with Linear Depth and Connectivity*, Phys. Rev. Lett., 120 (2018), p. 110501, <https://doi.org/10.1103/PhysRevLett.120.110501>.
- [22] E. KÖKCÜ, D. CAMPS, L. BASSMAN, J. K. FREERICKS, W. A. DE JONG, R. VAN BEEUMEN, ET AL., *Algebraic compression of quantum circuits for Hamiltonian evolution*, 2021, <https://arxiv.org/abs/2108.03282>.
- [23] E. KÖKCÜ, T. STECKMANN, J. K. FREERICKS, E. F. DUMITRESCU, AND A. F. KEMPER, *Fixed depth Hamiltonian simulation via Cartan decomposition*, 2021, <https://arxiv.org/abs/2104.00728>.
- [24] S. LLOYD, *Universal quantum simulators*, Science, 273 (1996), pp. 1073–1078, <https://doi.org/10.1126/science.273.5278.1073>.
- [25] G. H. LOW AND I. L. CHUANG, *Optimal Hamiltonian simulation by quantum signal processing*, Phys. Rev. Lett., 118 (2017), pp. 1–6, <https://doi.org/10.1103/PhysRevLett.118.010501>.
- [26] G. H. LOW AND I. L. CHUANG, *Hamiltonian simulation by qubitization*, Quantum, 3 (2019), p. 163, <https://doi.org/10.22331/q-2019-07-12-163>.
- [27] R. I. MCLACHLAN AND G. R. W. QUISPTEL, *Splitting methods*, Acta Numer., 11 (2002), pp. 341–434, <https://doi.org/10.1017/S0962492902000053>.
- [28] M. A. NIELSEN AND I. L. CHUANG, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, 2010, <https://doi.org/10.1017/CBO9780511976667>.
- [29] D. POULIN, A. QARRY, R. SOMMA, AND F. VERSTRAETE, *Quantum simulation of time-dependent Hamiltonians and the convenient illusion of Hilbert space*, Phys. Rev. Lett., 106 (2011), p. 170501, <https://doi.org/10.1103/PhysRevLett.106.170501>.
- [30] J. PRESKILL, *Quantum Computing in the NISQ era and beyond*, Quantum, 2 (2018), <https://doi.org/10.22331/q-2018-08-06-79>.
- [31] V. V. SHENDE, S. S. BULLOCK, AND I. L. MARKOV, *Synthesis of quantum-logic circuits*, IEEE TCAD, 25 (2006), pp. 1000–1010, <https://doi.org/10.1109/TCAD.2005.855930>.
- [32] B. D. SUTTON, *Computing the complete CS decomposition*, Numer. Algorithms, 50 (2009), pp. 33–65, <https://doi.org/10.1007/s11075-008-9215-6>.
- [33] B. D. SUTTON, *Stable computation of the CS decomposition: Simultaneous bidiagonalization*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 1–21, <https://doi.org/10.1137/100813002>.
- [34] M. SUZUKI, *General decomposition theory of ordered exponentials*, Proc. Jpn. Acad., Ser. B, 69 (1993), pp. 161–166, <https://doi.org/10.2183/pjab.69.161>.
- [35] M. THALHAMMER, *Convergence analysis of high-order time-splitting pseudospectral methods for nonlinear Schrödinger equations*, SIAM J. Numer. Anal., 50 (2012), <https://doi.org/10.1137/120866373>.
- [36] H. F. TROTTER, *On the product of semi-groups of operators*, Proc. Amer. Math. Soc., 10 (1959), <https://doi.org/10.1090/S0002-9939-1959-0108732-6>.
- [37] R. VAN BEEUMEN AND D. CAMPS, *F3C++*, 2021, <https://doi.org/10.5281/zenodo.5160875>, <https://github.com/QuantumComputingLab/f3cpp> (accessed 2021/08/04). Version 0.1.0.
- [38] R. VAN BEEUMEN AND D. CAMPS, *QCLAB++*, 2021, <https://doi.org/10.5281/zenodo.5160682>, <https://github.com/QuantumComputingLab/qclabpp> (accessed 2021/08/04). Version

- 0.1.2.
- [39] R. VANDEBRIL, *Chasing bulges or rotations? A metamorphosis of the QR-algorithm*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 217–247, <https://doi.org/10.1137/100809167>.
 - [40] R. VANDEBRIL AND D. S. WATKINS, *A generalization of the multishift QR algorithm*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 759–779, <https://doi.org/10.1137/11085219X>.
 - [41] J. J. VARTAINEN, M. MÖTIÖNEN, AND M. M. SALOMAA, *Efficient decomposition of quantum gates*, Phys. Rev. Lett., 92 (2004), pp. 1–4, <https://doi.org/10.1103/PhysRevLett.92.177902>.
 - [42] E. YOUNIS, K. SEN, K. YELICK, AND C. IANCU, *QFAST: conflating search and numerical optimization for scalable quantum circuit synthesis*, 2021, <https://arxiv.org/abs/2103.07093>.