Efficient product formulas for commutators and applications to quantum simulation

Yu-An Chen,[1,2,3,4,*] Andrew M. Childs,[4,5,6] Mohammad Hafezi,[2,4,7] Zhang Jiang,[1,†] Hwanmun Kim,[2] and Yijia Xu[2,4,8,‡]

[1]*Google Quantum AI, 340 Main Street, Venice, California 90291, USA*

[2]*Department of Physics and Joint Quantum Institute, NIST/University of Maryland, College Park, Maryland 20742, USA*

[3]*Condensed Matter Theory Center, University of Maryland, College Park, Maryland 20472 USA*

[4]*Joint Center for Quantum Information and Computer Science, University of Maryland, College Park, Maryland 20742, USA*

[5]*Department of Computer Science, University of Maryland, College Park, Maryland 20742, USA*

[6]*Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland 20742, USA*

[7]*Departments of Electrical and Computer Engineering and Institute for Research in Electronics and Applied Physics, University of Maryland, College Park, Maryland 20742, USA*

[8]*Institute for Physical Science and Technology, University of Maryland, College Park, Maryland 20742, USA*

(Received 9 December 2021; accepted 11 February 2022; published 10 March 2022)

We construct product formulas for exponentials of commutators and explore their applications. First, we directly construct a third-order product formula with six exponentials by solving polynomial equations obtained using the operator differential method. We then derive higher-order product formulas recursively from the third-order formula. We improve over previous recursive constructions, reducing the number of gates required to achieve the same accuracy. In addition, we demonstrate that the constituent linear terms in the commutator can be included at no extra cost. As an application, we show how to use the product formulas in a digital protocol for counterdiabatic driving, which increases the fidelity for quantum state preparation. We also discuss applications to quantum simulation of one-dimensional fermion chains with nearest- and next-nearest-neighbor hopping terms, and two-dimensional fractional quantum Hall phases.

DOI: 10.1103/PhysRevResearch.4.013191

I. INTRODUCTION

Product formulas approximate a desired unitary operator with a product of simpler operator exponentials. As a practical tool, product formulas are widely used in quantum simulation of condensed matter models and quantum chemistry problems [1–9], as well as quantum Monte Carlo and statistical physics problems [10–13]. The simplest example is the first-order Lie-Trotter product formula

$$e^{xA}e^{xB} = e^{x(A+B)} + O(x^2), \tag{1}$$

which approximates the sum of operators A and B. From the perspective of quantum simulation, this provides a way to approximate the time evolution of the Hamiltonian $H = A + B$ by multiplying elementary exponentials of the form $e^{-i\delta A}$ and $e^{-i\delta B}$ (which generally do not commute).

In addition to simulating Hamiltonian evolution of a linear combination of terms, one can also construct product formulas for commutators [14,15]. The simplest product formula for commutators is the second-order formula

$$S_2(x) := e^{xA}e^{xB}e^{-xA}e^{-xB} = e^{x^2[A,B]} + O(x^3). \tag{2}$$

Such commutator product formulas raise the possibility of simulating complicated unitaries on a quantum simulator using limited native gates: Given the time evolution of operators A and B, the time evolution of any linear combination of nested commutators involving A and B (i.e., the Lie algebra generated by A and B) can be simulated. Moreover, product formulas for commutators with arbitrary high order k, $\exp([A, B]x^2) + O(x^{k+1})$, have been constructed recursively [14,15].

Now we introduce terminology for product formulas. An mth-order product formula for a sum is a sequence of elementary exponentials of A and B that approximates $\exp[x(A + B)]$ to mth order in x:

$$e^{t_1A}e^{t_2B}e^{t_3A}e^{t_4B}\cdots = e^{x(A+B)} + O(x^{m+1}), \tag{3}$$

where the time interval for the ith elementary exponential is $t_i := \alpha_i x$ and $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \ldots$ are parameters that define the formula. Similarly, an mth-order commutator product formula is a sequence of elementary exponentials of A and B that approximates $e^{x^2[A,B]}$ to mth order:

$$e^{t_1A}e^{t_2B}e^{t_3A}e^{t_4B}\cdots = e^{x^2[A,B]} + O(x^{m+1}). \tag{4}$$

Notice that the prefactor of $[A, B]$ is x^2 since it arises from terms quadratic in $t_i = \alpha_i x$. Both sum and commutator product

*yuanchen@umd.edu
†jiangzhang@google.com
‡yijia@umd.edu

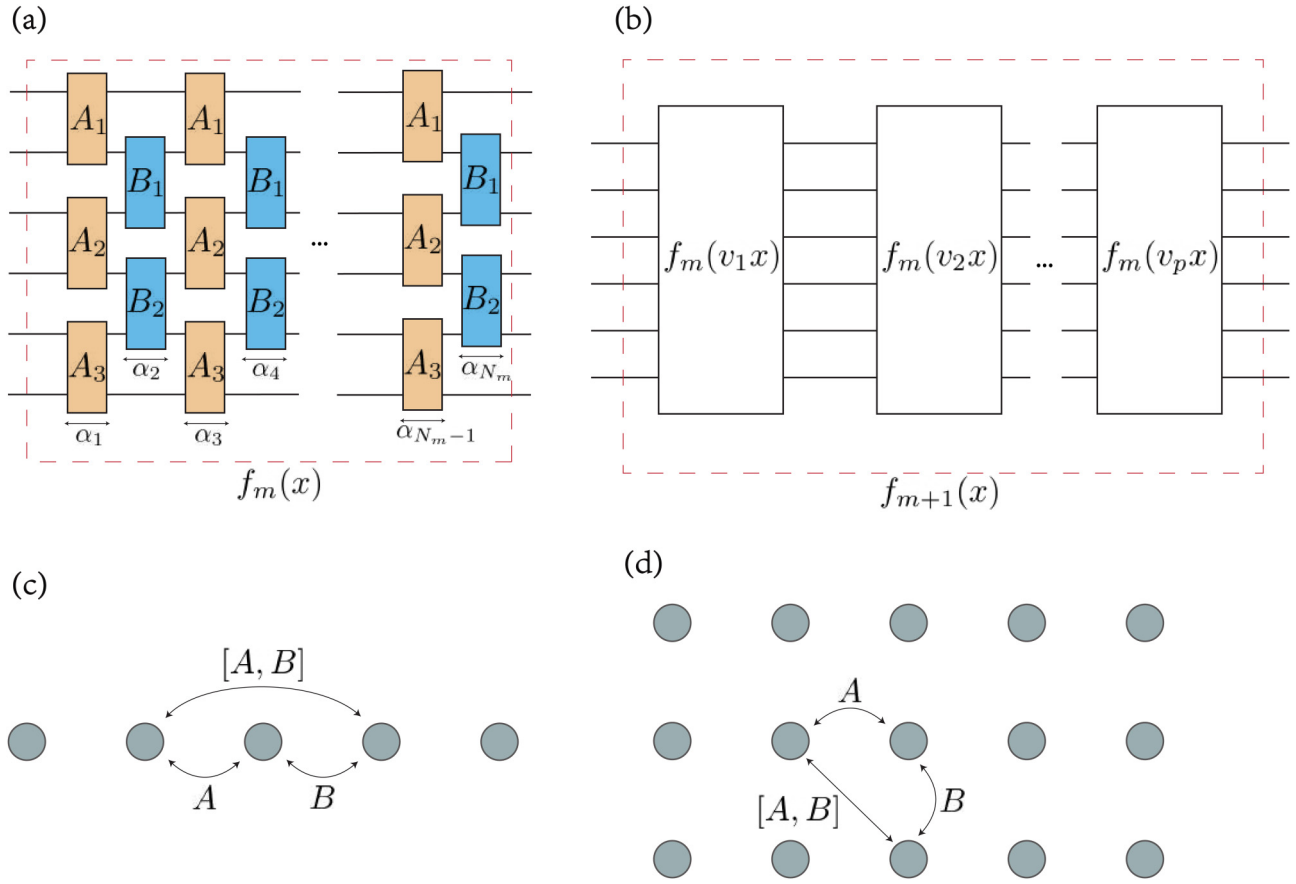

FIG. 1. (a) Exact construction of an mth-order N_m-gate product formula. e^{At} and $e^{Bt'}$ are native gates, and the circuit inside the box is the circuit representation of the mth-order product formula $f_m(x)$. In practice, operator $A = \sum_i A_i$ or $B = \sum_i B_i$ is decomposed as a sum of commuting terms (no overlap between each other). Considering the native gate sets on practical quantum computers, here we use a two-qubit gate as an example. (b) Construction of an $(m+1)$st-order product formula from an mth-order formula by a p-copy recursive formula. The starting point is the mth-order product formula $f_m(x)$, and the integrated circuit inside the box is the $(m+1)$st-order product formula $f_{m+1}(x)$, where each component can be $f_m(v_ix)$ or $f_m^{-1}(v_ix)$ (not shown in the figure). (c) and (d) Generated next-nearest-neighbor interaction by the commutator between two nearest-neighbor terms A and B in the 1d chain and the 2d square lattice, e.g., Eqs. (5) and (6).

formulas can be represented as quantum circuits, as shown in Fig. 1(a). In general, directly determining suitable coefficients α_i is difficult. Reference [13] provides an operator differential method for computing the coefficients of commutator product formulas, but it is hard to solve the resulting polynomial equations.

High-order product formulas can be constructed recursively. In this approach, we choose an invertible product formula $f_m(x)$ as the base formula and recursively increase accuracy with some prescribed sequence of terms of the form $f_m(v_ix)$ and $f_m(v_ix)^{-1}$ for appropriate coefficients v_i. By iterating this procedure, we can produce an arbitrarily high-order approximation of the target exponential. Thus the recursive method can be viewed as a "product formula of product formulas" where a lower-order product formula is the elementary unit. We call such a recursive formula a *p copy* if it uses p elementary formulas f_m and f_m^{-1} to improve the order by 1. We also consider recursive constructions that use q elementary formulas f_m and f_m^{-1} to improve the order by 2, which we call a $\sqrt{q}$-copy recursive formula.

See Table I for a comparison of directly constructed product formulas and recursive formulas. Given an mth-order product formula f_m that uses N_m elementary gates, the recursive construction gives an $(m+1)$st-order product formula f_{m+1} with $N_{m+1} = pN_m - O(p)$ gates [16], as shown in Fig. 1(b). Starting from an mth-order product formula, we can apply the recursive formula k times to get an $(m+k)$th-order product formula.

For practical quantum computation, it is essential to find product formulas that are as efficient as possible. Previous research [14,15] starts from the second-order commutator product formula equation (2) and uses different recursive methods to improve accuracy. The parameters of these previous recursive constructions are summarized at the top of Table II.

In general, the number of gates in any mth-order product formula is exponential in m [17]. However, distinct methods have different constants that affect the cost of product formulas in practice. In principle, high-order formulas could be constructed directly. However, solving algebraic equations to determine such a formula can be challenging, especially at high orders. Recursive constructions can straightforwardly build higher-order product formulas, potentially at the cost of worse performance than a direct construction.

TABLE I. Comparison between direct and recursive product formulas.

	N_m-gate mth-order product formula	p-copy recursive product formula
Explicit form	$f_m(x) = e^{\alpha_1 xA} e^{\alpha_2 xB} e^{\alpha_3 xA} e^{\alpha_4 xB} \cdots e^{\alpha_{N_m} xB}$	$f_{m+1}(x) = f_m(v_1 x)^{\pm 1} f_m(v_2 x)^{\pm 1} \cdots f_m(v_p x)^{\pm 1}$
Elementary unit	native gates e^{At}, e^{Bt}	a product formula $f_m(x)$ and inverse formula $f_m(x)^{-1}$
Number of gates	N_m	$pN_m - O(p)$
Error	$O(x^{m+1})$	$O(x^{m+2})$
Parameters	$\alpha_1, \alpha_2, \ldots, \alpha_n$	$v_1, v_2, \ldots, v_p$

Finally, we introduce some basic applications of commutator product formulas. Three-spin interactions can naturally emerge from commutators of two-spin terms. For example,

$$\left[\sigma_x^i \sigma_x^j, \sigma_z^j \sigma_z^k\right] = -2i\sigma_x^i \sigma_y^j \sigma_z^k. \tag{5}$$

More generally, we can construct multispin operators from lower-order terms. Furthermore, the commutator between two bosonic or fermionic nearest-neighbor hopping terms is a next-nearest-neighbor hopping term:

$$[a_i^\dagger a_j + a_j^\dagger a_i, a_j^\dagger a_k + a_k^\dagger a_j] = a_i^\dagger a_k - a_k^\dagger a_i, \tag{6}$$

where $a_i, a_i^\dagger$ are bosonic or fermionic creation and annihilation operators that obey $[a_i, a_j^\dagger]_\pm = \delta_{i,j}$. This technique can be used to generate complicated interactions from simple nearest-neighbor interactions. In Fig. 1(c), we show the construction of next-nearest-neighbor hopping in a one-dimensional (1d) system using a commutator. We show a similar approach to next-nearest-neighbor hopping for a 2d system in Fig. 1(d).

Summary of results

In this paper, we use the operator differential method [13] to construct product formulas that combine sums and commutators. Specifically, as described in Sec. II and Appendix A, we construct a formula that implements

$$e^{x(A+B)+Rx^2[A,B]} + O(x^4) \tag{7}$$

for arbitrary $R \in \mathbb{R}$, using six exponentials of A and B.

In the large-R limit, our product formula reduces to the pure commutator formula

$$\begin{aligned} S_3(x) &:= e^{\frac{\sqrt{5}-1}{2}xA} e^{\frac{\sqrt{5}-1}{2}xB} e^{-xA} e^{-\frac{\sqrt{5}+1}{2}xB} e^{\frac{3-\sqrt{5}}{2}xA} e^{xB} \\ &= e^{x^2[A,B]} + O(x^4), \end{aligned} \tag{8}$$

which reduces the error by one order in x compared with the group commutator formula equation (2), giving substantially better performance in practice.

In a recursive construction of higher-order product formulas, a good base formula can have significant impact. We show that the third-order commutator product formula $S_3(x)$ can improve recursive methods. We numerically check that previous recursive constructions can perform better when using $S_3(x)$ instead of $S_2(x)$ as the base formula. In particular, this change significantly reduces the total gate count required to achieve a fixed error.

In addition to a better base formula, we improve the recursive method. We first modify the Childs-Wiebe $\sqrt{6}$-copy formula (Theorem 2 in Ref. [15]), which uses six instances of an even-order formula f_{2k} to increase its order by 2. This construction first applies a two-copy recursive formula to increase the order from $2k$ to $2k+1$ and then applies a three-copy recursive formula to get a $(2k+2)$nd-order product formula. We observe that these two steps can be decomposed. In particular, if we start with an odd-order product formula, we can apply the three-copy recursive formula first and then apply the two-copy one. This modified Childs-Wiebe $\sqrt{6}$-copy formula is denoted V in Table II.

We further propose $\sqrt{4}$-copy, $\sqrt{5}$-copy, and $\sqrt{10}$-copy recursive formulas that use $(4N_m - 3)$, $(5N_m - 4)$, and $(10N_m - 4)$ gates, respectively, to generate $(m+2)$nd-order formulas from an N_m-gate mth-order formula. Note that using fewer gates to achieve a given order is not necessarily better, since constant factors in the error terms can significantly affect performance. Indeed, using numerical simulations, we demonstrate that our $\sqrt{10}$-copy recursive formula requires the fewest gates to reach the same accuracy.

In summary, we find that (1) the third-order product formula $S_3(x)$ [Eq. (8)] performs better than the standard choice $S_2(x)$ [Eq. (2)], serving as a better base formula for all recursive methods in Table II, and (2) other recursive formulas

TABLE II. Comparison of recursive formulas. The first three formulas are previous approaches. The remaining formulas are described in this paper.

Recursive formula	Total number of copies	Accuracy improvement	Number of gates
Jean-Koseleff [14, Lemma 7]	3	$O(x^{m+1}) \to O(x^{m+2})$	$N_{m+1} = 3N_m - 2$
Childs-Wiebe (five copy) [15, Lemma 7]	5	$O(x^{m+1}) \to O(x^{m+2})$	$N_{m+1} = 5N_m - 2$
Childs-Wiebe ($\sqrt{6}$ copy) [15, Theorem 2]	6	$O(x^{2k+1}) \to O(x^{2k+3})$	$N_{2k+2} = 6N_{2k} - 2$
Q ($\sqrt{4}$ copy)	4	$O(x^{m+1}) \to O(x^{m+3})$	$N_{m+2} = 4N_m - 3$
W ($\sqrt{5}$ copy)	5	$O(x^{m+1}) \to O(x^{m+3})$	$N_{m+2} = 5N_m - 4$
V ($\sqrt{6}$ copy)	6	$O(x^{2k}) \to O(x^{2k+2})$	$N_{2k+1} = 6N_{2k-1} - 4$
$\mathcal{G}$ ($\sqrt{10}$ copy)	10	$O(x^{m+1}) \to O(x^{m+3})$	$N_{m+2} = 10N_m - 4$

can offer better performance, with the $\sqrt{10}$-copy formula performing the best of those we study.

We present two concrete applications of our product formulas. The first application is in the context of the counterdiabatic driving, a method that was originally proposed for analog quantum computation using nested commutators [20–23]. In the context of digital quantum state preparation, we demonstrate that our product formulas can generate the commutator terms required for counterdiabatic driving. This additional term increases the fidelity of the final state without increasing the number of gates. To illustrate the efficiency of our approach, we consider state preparation for spins as an example.

The second application is in the context of the quantum simulation. We show that a one-dimensional fermion chain with next-nearest-neighbor hopping terms and a model of two-dimensional fractional quantum Hall phases can be naturally simulated using our method. In both cases, we use nearest-neighbor hopping terms to generate next-nearest-neighbor hopping terms, which break time-reversal symmetry.

The remainder of the paper is organized as follows. In Sec. II, we use the operator differential method to construct the third-order commutator product formula. In addition, we propose a product formula for combined sums and commutators in Sec. II B. Then we derive recursive product formulas for commutators in Sec. III. We present numerical simulations of the third-order product formula and recursive constructions in Sec. IV. Next, in Sec. V A, we demonstrate that the formula proposed in this paper can implement counterdiabatic driving and generate next-nearest-neighbor interactions from nearest-neighbor terms. We show how to simulate a 1d fermion chain with next-nearest-neighbor hopping terms in Sec. V B and fractional quantum Hall phases (the Kapit-Mueller model) in Sec. V C. Finally, we discuss open questions and future directions in Sec. VI.

II. THIRD-ORDER PRODUCT FORMULA

In this section, we introduce the third-order exponential product formula, i.e., a formula with error $O(x^4)$ where each elementary exponential has time proportional to x. We first present a formula for the pure commutator $[A, B]$ and then present a product formula that includes both the sum $A + B$ and commutator $[A, B]$.

We derive these product formulas using the operator differential method [13], as detailed in Appendix A. In particular, we find the following general expression for a six-gate product formula:

$$e^{p_1 xA} e^{p_2 xB} e^{p_3 xA} e^{p_4 xB} e^{p_5 xA} e^{p_6 xB} = \exp[\Phi(x)], \tag{9}$$

with

$$\begin{aligned}\Phi(x) = x(lA + mB) + \frac{x^2}{2}(lm - 2q)[A, B] \\ + \frac{x^3}{6}\left(\left(\frac{l^2 m}{2} - 3r\right)[A, [A, B]]\right. \\ \left. + \left(\frac{m^2 l}{2} - 3s\right)[B, [B, A]]\right) + O(x^4),\end{aligned} \tag{10}$$

where

$$\begin{aligned} l &:= p_1 + p_3 + p_5, \\ m &:= p_2 + p_4 + p_6, \\ q &:= p_2 p_3 + p_2 p_5 + p_4 p_5, \\ r &:= p_1 p_2 p_3 + p_1 p_2 p_5 + p_1 p_4 p_5 + p_3 p_4 p_5, \\ s &:= p_2 p_3 p_4 + p_2 p_3 p_6 + p_2 p_5 p_6 + p_4 p_5 p_6. \end{aligned} \tag{11}$$

Thus an arbitrary six-gate product formula $\Phi(x)$ can be reparametrized by l, m, q, r, s, which are functions of $p_1, \ldots, p_6$. By fine-tuning the parameters $p_1, \ldots, p_6$ (or equivalently, l, m, q, r, s), we can obtain the desired $\Phi(x)$. Here we focus on two cases: a third-order commutator product formula and a third-order product formula for a combination of the sum and commutator.

To construct a third-order product formula for the commutator, we solve for $p_1, \ldots, p_6$ so that

$$\begin{aligned} l = m = 0, \\ lm - 2q \neq 0, \\ \frac{l^2 m}{2} - 3r = \frac{m^2 l}{2} - 3s = 0. \end{aligned} \tag{12}$$

These conditions ensure that $\Phi(x)$ only involves the commutator $[A, B]$ and terms that are $O(x^4)$.

Similarly, for the third-order product formula for both sum and commutator, we would like to keep both the linear and (non-nested) commutator terms. Hence we should find $p_1, \ldots, p_6$ so that

$$\begin{aligned} l = m \neq 0, \\ lm - 2q \neq 0, \\ \frac{l^2 m}{2} - 3r = \frac{m^2 l}{2} - 3s = 0. \end{aligned} \tag{13}$$

We discuss the details in Sec. II A (pure commutator) and Sec. II B (sum and commutator).

A. Pure commutator

We now derive the six-gate third-order product formula for commutators:

$$\begin{aligned} S_3(x) &:= \exp\left(\tfrac{\sqrt{5}-1}{2} xA\right) \exp\left(\tfrac{\sqrt{5}-1}{2} xB\right) \exp(-xA) \\ &\quad \times \exp\left(-\tfrac{\sqrt{5}+1}{2} xB\right) \exp\left(\tfrac{3-\sqrt{5}}{2} xA\right) \exp(xB) \\ &= \exp(x^2[A, B] + O(x^4)). \end{aligned} \tag{14}$$

This formula can be checked by using the Taylor series of each term up to order x^3 to show that the constant, x, x^2, and x^3 terms on both sides agree. In general, by Eq. (10), the product formula

$$e^{p_1 xA} e^{p_2 xB} e^{p_3 xA} e^{p_4 xB} e^{p_5 xA} e^{p_6 xB} = \exp\left(x^2[A, B] + O(x^4)\right) \tag{15}$$

holds if p_1, p_2, p_3, p_4, p_5, p_6 satisfy the following polynomial equations resulting from Eq. (12):

$$\begin{aligned} l = m = r = s = 0, \\ q = p_2 p_3 + p_2 p_5 + p_4 p_5 = -1. \end{aligned} \tag{16}$$

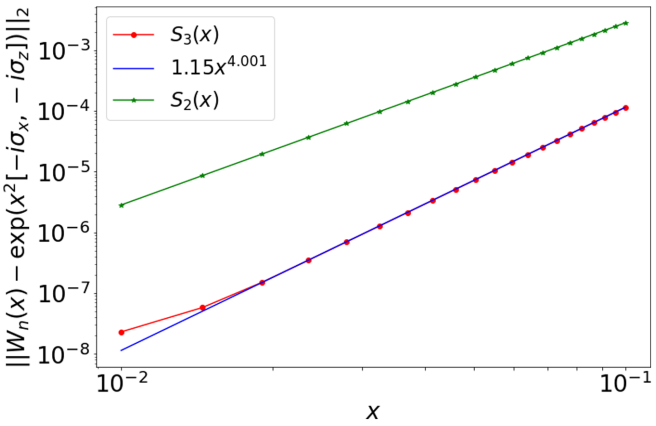

FIG. 2. Error scaling for $S_2(x)$ and $S_3(x)$ with $A = -i\sigma_x$, $B = -i\sigma_z$. The error of $S_3(x)$ is smaller than that of $S_2(x)$ by roughly 10^{-2}.

Equation (14) is a particular solution of the above equations [24]. Figure 2 shows the empirical error scaling behavior of $S_3(x)$ for a one-qubit example. The error exponent obtained by fitting the data points in the interval $2 \times 10^{-2} \leqslant x \leqslant 10^{-1}$ is 4.001, in good agreement with theory.

B. Sum and commutator

In this section, we consider the product formula for both sum and commutator. For arbitrary $R \in \mathbb{R}$, we want to find a set of parameters $p_1(R), \ldots, p_6(R)$ such that

$$\Phi(x) = x(A+B) + Rx^2[A,B] + O(x^4). \tag{17}$$

This is equivalent to solving the equations given by Eq. (13):

$$\begin{aligned} l &= m = 1, \\ q &= -R + \frac{1}{2}, \\ r &= s = \frac{1}{6}. \end{aligned} \tag{18}$$

For a specific value of R, we can solve these equations numerically and obtain a third-order product formula with the exponent equation (17). In general, these equations are difficult to solve analytically, but one can find an approximate solution for large R:

$$\begin{aligned} p_1 &= (g-1)\sqrt{R+\frac{1}{2}}, \quad p_2 = (g-1)\sqrt{R+\frac{1}{2}} + 1, \\ p_3 &= -\sqrt{R+\frac{1}{2}} + 1, \quad p_4 = -g\sqrt{R+\frac{1}{2}}, \\ p_5 &= (2-g)\sqrt{R+\frac{1}{2}}, \quad p_6 = \sqrt{R+\frac{1}{2}}, \end{aligned} \tag{19}$$

where $g := \frac{\sqrt{5}+1}{2}$. This choice corresponds to

$$\begin{aligned} l &= m = 1, \\ q &= -R + \frac{1}{2}, \\ r &= -(g-1)\left(R + \frac{1}{2} - \sqrt{R+\frac{1}{2}}\right), \\ s &= (g-1)\left(R + \frac{1}{2} - \sqrt{R+\frac{1}{2}}\right). \end{aligned} \tag{20}$$

While this does not satisfy $r = s = \frac{1}{6}$, the leading order of r and s is $O(R)$ [whereas for a general choice of $\{p_i\}$, the leading order is $O(R^{\frac{3}{2}})$]. Substituting Eq. (20) into Eq. (10), we find that the linear term vanishes, the quadratic term is

$$\frac{lm-2q}{2} x^2[A,B] = Rx^2[A,B], \tag{21}$$

and the x^3 term is

$$\begin{aligned} &\frac{x^3}{6}\left(\left(\frac{l^2 m}{2} - 3r\right)[A,[A,B]] + \left(\frac{m^2 l}{2} - 3s\right)[B,[B,A]]\right) \\ &\quad = x^3 O(R). \end{aligned} \tag{22}$$

From Eq. (10), we have constructed

$$\begin{aligned} f_R(x) = \exp\big[&x(A+B) + Rx^2[A,B] + x^3 O(R) \\ &+ x^4 O(R^2) + \cdots + x^k O(R^{\frac{k}{2}}) + \cdots\big]. \end{aligned} \tag{23}$$

Notice that the coefficient in front of the x^k term for $k \geqslant 4$ contains products of $p_{i_1} p_{i_2} \cdots p_{i_k}$, where each p_{i_j} is $O(R^{\frac{1}{2}})$.

We then use the third-order product formula to implement a new gate from existing gates. Assume that we can perform gates of the form $e^{\theta_1 A}$ and $e^{\theta_2 B}$ for $\theta_1, \theta_2 \in \mathbb{R}$ and our goal is to perform

$$e^{\alpha(A+B)+\beta[A,B]} \tag{24}$$

for some desired $\alpha, \beta \in \mathbb{R}$. First, we pick a large integer n such that $x := \frac{\alpha}{n}$ is small and $R := \frac{\beta n}{\alpha^2}$ is large. Using the function equation (23) constructed above, we have

$$\begin{aligned} f_R\left(\frac{\alpha}{n}\right) = \exp\Bigg[&\frac{\alpha}{n}(A+B) + \frac{\beta}{n}[A,B] + O\left(\frac{\alpha\beta}{n^2}\right) \\ &+ \cdots + O\left(\frac{\beta^{\frac{k}{2}}}{n^{\frac{k}{2}}}\right) + \cdots\Bigg]. \end{aligned} \tag{25}$$

Repeating this function n times gives the desired gate:

$$f_{\frac{\beta n}{\alpha^2}}\left(\frac{\alpha}{n}\right)^n = \exp\left[\alpha(A+B) + \beta[A,B] + O\left(\frac{\alpha\beta+\beta^2}{n}\right)\right]. \tag{26}$$

This implementation uses $6n$ gates and achieves error $O(\frac{1}{n})$.

In the limit $\alpha \to 0$, β constant, and $n \gg 1$, Eq. (25) converges to

$$\begin{aligned} &\exp\left((g-1)\sqrt{\frac{\beta}{n}}A\right)\exp\left((g-1)\sqrt{\frac{\beta}{n}}B\right)\exp\left(-\sqrt{\frac{\beta}{n}}A\right) \\ &\quad\times \exp\left(-g\sqrt{\frac{\beta}{n}}B\right)\exp\left((2-g)\sqrt{\frac{\beta}{n}}A\right)\exp\left(\sqrt{\frac{\beta}{n}}B\right) \\ &\quad = \exp\left(\frac{\beta}{n}[A,B] + O\left(\frac{\beta^2}{n^2}\right)\right), \end{aligned} \tag{27}$$

where $g = \frac{\sqrt{5}+1}{2}$, which reduces to the pure commutator formula in Sec. II A. Notice that the error has the same order with or without the sum $A+B$, which means that there is no extra cost to simulate the sum along with the commutator.

In Sec. V, we use Eq. (26) to generate new Hamiltonian terms from existing ones, such as next-nearest-neighbor (NNN) hopping terms from nearest-neighbor (NN) hopping terms. There is another useful formula that can easily be derived from Eq. (25):

$$\exp\left(\frac{\alpha}{n}C\right)f_R\left(\frac{\alpha}{n}\right) = \exp\left(\frac{\alpha}{n}(A+B+C)+\frac{\beta}{n}[A,B]+O\left(\frac{1}{n^2}\right)\right), \quad (28)$$

or equivalently,

$$\left(\exp\left(\frac{\alpha}{n}C\right)f_R\left(\frac{\alpha}{n}\right)\right)^n = \exp\left(\alpha(A+B+C)+\beta[A,B]+O\left(\frac{1}{n}\right)\right), \quad (29)$$

which uses $7n$ gates and has error $O(1/n)$.

III. RECURSIVE FORMULAS

In this section, we introduce the recursive construction of higher-order product formulas. We first focus on the pure commutator formula, where we improve over previous procedures [14,15]. Then we discuss recursive formulas for both sum and commutator, following the same strategy as the recursive formula for the sum alone [13,25,26].

A. Pure commutator

In this section, we introduce recursive formulas that use 4, 5, 6, or 10 copies of an nth-order formula to generate an $(n+2)$nd-order formula. To begin, assume that we have an nth-order formula for the commutator, of the form

$$f_n(x) = \exp(x^2[A,B]) + C_n x^{n+1} + D_n x^{n+2} + O(x^{n+3}) \quad (30)$$

for some coefficients $C_n, D_n \in R$. As in previous recursive constructions, we make essential use of inverse product formulas. Given an nth-order product formula $f_n(x)$, its inverse formula is $f_n(x)^{-1}$, where

$$f_n(x)^{-1} f_n(x) = 1. \quad (31)$$

Since $f_n(x)$ is a product of elementary exponentials

$$f_n(x) = e^{p_1 xA} e^{p_2 xB} e^{p_3 xA} \cdots e^{p_k xB}, \quad (32)$$

its inverse is simply

$$f_n(x)^{-1} = (e^{p_1 xA} e^{p_2 xB} e^{p_3 xA} \cdots e^{p_k xB})^{-1} = e^{-p_k xB} \cdots e^{-p_3 xA} e^{-p_2 xB} e^{-p_1 xA}. \quad (33)$$

Notice that we include coefficients C_n and D_n to keep track of the x^{n+1} and x^{n+2} terms, respectively. From $f_n(x)$, we can construct other product formulas:

$$\begin{aligned} f_n^{-1}(x) &= \exp(-x^2[A,B]) - C_n x^{n+1} - D_n x^{n+2}, \\ f_n(-x) &= \exp(x^2[A,B]) - (-1)^n C_n x^{n+1} + (-1)^n D_n x^{n+2}, \\ f_n^{-1}(-x) &= \exp(-x^2[A,B]) + (-1)^n C_n x^{n+1} - (-1)^n D_n x^{n+2}, \end{aligned} \quad (34)$$

where we omit the $O(x^{n+3})$ error term for brevity. We use $f_n(x)$ and Eq. (34) as building blocks for higher-order product formulas.

In particular, if $n = 2k$ is even, there is a recursive formula that increases the order of the product formula by 1 using only two copies of the product formula f_n [15, Corollary 3]:

$$f_{2k+1}(x) := f_{2k}\left(\frac{x}{\sqrt{2}}\right) f_{2k}\left(-\frac{x}{\sqrt{2}}\right) = \exp(x^2[A,B]) + O(x^{2k+2}). \quad (35)$$

For general n, there are two previously established ways to increase the order by 1.

(1) One way is the Jean-Koseleff formula [14]:

$$f_{n+1}(x) = \exp\left(x^2[A,B] + O(x^{n+2})\right) = \begin{cases} f_n(tx) f_n(sx) f_n(tx) & \text{if } n \text{ is even} \\ f_n(ux) f_n(vx)^{-1} f_n(ux) & \text{if } n \text{ is odd,} \end{cases} \quad (36)$$

with $t = (2+2^{2/(n+1)})^{-1/2}$, $s = -2^{1/(n+1)}t$, $u = (2-2^{2/(n+1)})^{-1/2}$, and $v = 2^{1/(n+1)}u$.

(2) The other way is the Childs-Wiebe (five-copy) formula [15]:

$$f_{n+1}(x) = \exp\left(x^2[A,B] + O(x^{n+2})\right) = f_n(vx)^2 f_n(\mu x)^{-1} f_n(vx)^2, \quad (37)$$

with $\mu = (4s_n)^{1/2}$, $v = (1/4+\sigma)^{1/2}$, and $\sigma = \frac{4^{\frac{2}{n+1}}}{4(4-4^{\frac{2}{n+1}})}$.

1. $\sqrt{6}$-copy recursive formula

Theorem 2 of Ref. [15] defines a $\sqrt{6}$-copy recursive construction that improves the order of an even-order product formula $f_{2k}(x)$ by 2 using six copies of $f_{2k}(x)$ and $f_{2k}(x)^{-1}$. We observe that this construction first applies the two-copy formula equation (35) and then applies the Jean-Koseleff formula equation (36). Alternatively, we can consider these two steps independently and combine them in different ways. In particular, given an odd-order product formula, we can first apply the the Jean-Koseleff formula and then apply the two-copy formula.

Here, we explicitly describe this alternative $\sqrt{6}$-copy recursion for odd-order product formulas. Let $V_n(x)$ be a product formula that approximates $\exp(x^2[A,B])$ with error $O(x^{n+1})$ for odd n.

The two-step recursive relation has the form

$$V_{n+2}(x) = V_n\left(\frac{ux}{\sqrt{2}}\right) V_n\left(\frac{vx}{\sqrt{2}}\right)^{-1} V_n\left(\frac{ux}{\sqrt{2}}\right) \times V_n\left(\frac{-ux}{\sqrt{2}}\right) V_n\left(\frac{-vx}{\sqrt{2}}\right)^{-1} V_n\left(\frac{-ux}{\sqrt{2}}\right), \quad (38)$$

where $u = (2-2^{2/(n+1)})^{-1/2}$ and $v = 2^{1/(n+1)}u$. Here, the first three terms and the last three terms are $(n+1)$st-order formulas, which combine to give an $(n+2)$nd-order formula. This construction increases the order by 2 using six copies of V_n and V_n^{-1}, so we call it a $\sqrt{6}$-copy recursion.

This approach can applied to our third-order formula $S_3(x)$ to get higher-order formulas. Letting N_n^V denote the number

of gates in the nth-order formula, we have

$$N_{n+1}^V = \begin{cases} 3N_n^V - 2 & \text{if } n \text{ is odd} \\ 2N_2^V & \text{if } n \text{ is even.} \end{cases} \tag{39}$$

Starting from $N_3 = 6$, this gives

$$N_{2k+1}^V = \frac{1}{15}(13 \cdot 6^k + 12). \tag{40}$$

2. $\sqrt{10}$-copy recursive formula

Let $\mathcal{G}_n(x)$ be an invertible product formula that approximates $\exp(x^2[A,B])$ with error $O(x^{n+1})$. If n is odd, the Childs-Wiebe (five-copy) formula, Eq. (37), can be used to increase its order by 1. If n is even, we can apply Eq. (35) to increase the order by 1 using two copies of $\mathcal{G}_n$. Overall, we use ten copies of $\mathcal{G}_n$ and $\mathcal{G}_n^{-1}$ to increase the order by 2. Therefore $\mathcal{G}_n(x)$ is a $\sqrt{10}$-copy product formula.

Let $N_n^{\mathcal{G}}$ denote the number of gates in the nth-order formula. For odd $n = 2k+1$, we have

$$N_{2k+2}^{\mathcal{G}} = 5N_{2k+1}^{\mathcal{G}} - 2, \tag{41}$$

and for even $n = 2k$, we have

$$N_{2k+1}^{\mathcal{G}} = 2N_{2k}^{\mathcal{G}}. \tag{42}$$

Combining Eqs. (41) and (42), we have

$$N_{2k+3}^{\mathcal{G}} = 10N_{2k+1}^{\mathcal{G}} - 4. \tag{43}$$

Starting from the base formula with $N_3 = 6$, we have

$$N_{2k+1}^{\mathcal{G}} = \frac{1}{9}(5 \cdot 10^k + 4). \tag{44}$$

3. $\sqrt{5}$-copy recursive formula

We now consider the product

$$\begin{aligned} &W_n(-s'x)W_n^{-1}(x)W_n(sx)W_n^{-1}(-x)W_n(-s'x) \\ &= \exp\left((s^2 + 2s'^2 - 2)x^2[A,B]\right) \\ &\quad + (s^{n+1} + 2s'^{n+1} - 2)C_n x^{n+1} \\ &\quad + (s^{n+2} - 2s'^{n+2})D_n x^{n+2} + O(x^{n+3}). \end{aligned} \tag{45}$$

We first choose $s' = 2^{-\frac{1}{n+2}}s$ such that $s^{n+2} - 2s'^{n+2} = 0$. To eliminate the coefficient $(s^{n+1} + 2s'^{n+1} - 2)$, we have $s^{n+1} + 2s'^{n+1} = (1 + 2^{\frac{1}{n+2}})s^{n+1} = 2$. Therefore we choose $s = (\frac{2}{1+2^{\frac{1}{n+2}}})^{\frac{1}{n+1}}$. Then we define a new variable

$$\begin{aligned} x' &:= x\sqrt{(s^2 + 2s'^2 - 2)} \\ &= x\sqrt{\left(\frac{2}{1+2^{\frac{1}{n+2}}}\right)^{\frac{2}{n+1}}(1 + 2^{\frac{n}{n+2}}) - 2} \\ &=: rx. \end{aligned} \tag{46}$$

We can check that $r > 0$ for $n > 1$. Finally, we get the $(n+2)$nd-order formula

$$W_{n+2}(x') = W_n(-s'\tfrac{x'}{r})W_n^{-1}(\tfrac{x'}{r})W_n(s\tfrac{x'}{r})W_n^{-1}(-\tfrac{x'}{r})W_n(-s'\tfrac{x'}{r}). \tag{47}$$

Letting N_n^W denote the number of gates in W_n, we have the recursive relation

$$N_{n+2}^W = 5N_n^W - 4. \tag{48}$$

With $N_3 = 6$, we have

$$N_{2k+1}^W = 5^k + 1. \tag{49}$$

4. $\sqrt{4}$-copy recursive formula

We now discuss a way to use only four copies of an nth-order product formula to generate an $(n+2)$nd-order product formula. Let $Q_n(x)$ be an invertible product formula that approximates $\exp(x^2[A,B])$ with error $O(x^{n+1})$. Consider the following product:

$$\begin{aligned} &Q_n(ax)Q_n^{-1}(bx)Q_n(cx)Q_n^{-1}(dx) \\ &\quad = \exp\left((a^2 - b^2 + c^2 - d^2)x^2[A,B]\right) \\ &\qquad + (a^{n+1} - b^{n+1} + c^{n+1} - d^{n+1})C_n x^{n+1} \\ &\qquad + (a^{n+2} - b^{n+2} + c^{n+2} - d^{n+2})D_n x^{n+2} + O(x^{n+3}). \end{aligned} \tag{50}$$

To produce a formula of order $n+2$, we want to find a, b, c, d satisfying

$$\begin{aligned} a^{n+1} - b^{n+1} + c^{n+1} - d^{n+1} &= 0, \\ a^{n+2} - b^{n+2} + c^{n+2} - d^{n+2} &= 0, \\ a^2 - b^2 + c^2 - d^2 &\neq 0. \end{aligned} \tag{51}$$

If a solution exists, we can define a new variable $x' = sx$, with $s := \sqrt{|a^2 - b^2 + c^2 - d^2|}$, to find the $(n+2)$nd-order formula

$$\begin{aligned} &Q_n(\tfrac{a}{s}x')Q_n^{-1}(\tfrac{b}{s}x')Q_n(\tfrac{c}{s}x')Q_n^{-1}(\tfrac{d}{s}x') \\ &= \begin{cases} Q_{n+2}(x') & \text{if } a^2 - b^2 + c^2 - d^2 > 0 \\ Q_{n+2}^{-1}(x') & \text{if } a^2 - b^2 + c^2 - d^2 < 0. \end{cases} \end{aligned} \tag{52}$$

Let N_n^Q be the number of gates of the nth-order formula. The number of gates in such a formula satisfies

$$N_{2k+3}^Q = 4N_{2k+1}^Q - 3. \tag{53}$$

With $N_3 = 6$, we have

$$N_{2k+1}^Q = 5 \cdot 4^{k-1} + 1. \tag{54}$$

We can take $a = 1$, $b = 2$, and numerically solve Eq. (51) to find c and d. Table III presents numerical solutions for $n = 3, 5, 7, 9, 11$. We prove in Appendix B that a solution exists for general n.

B. Sum and commutator

We also construct a recursive formula for the product formula equation (25) for a linear combination of a sum and a commutator, using the same idea as the $\sqrt{6}$-copy approach described above. Suppose that we have an mth-order product formula of the form

$$f_{R,m}(x) = \exp\left(x(A+B) + \frac{x\beta}{\alpha}[A,B]\right) + C_m x^{m+1} + O(x^{n+2}). \tag{55}$$

TABLE III. Numerical solutions of Eq. (51) for $n = 3, 5, 7, 9, 11$.

	$n=3$	$n=5$	$n=7$	$n=9$	$n=11$
a	1	1	1	1	1
b	2	2	2	2	2
c	1.982590733	1.996950166	1.999411381	1.999880034	1.999974677
d	−0.8190978288	−0.8642318466	−0.8911860667	−0.9091844711	−0.9220693131
$a^2-b^2+c^2-d^2$	0.2597447625	0.2409130177	0.2034332678	0.1729037481	0.1496868917

In this formula, the commutator term scales with x instead of x^2. While the commutator term is $x^2R[A,B]$ in Eq. (23), it becomes $\frac{x\beta}{\alpha}[A,B]$ with the choice of large $R=\frac{\beta}{\alpha x}$. Since $A+B$ and $[A,B]$ are both first-order terms, the recursive formula for sum and commutator should be similar to the recursive formula for sum. Here, we use Suzuki's method [26] to construct the recursive product formula for sum and commutator.

If m is even, then we consider the three-copy sequence

$$f_{R,m}(ax)f_{R,m}^{-1}(bx)f_{R,m}(ax) = \exp\left((2a-b)x(A+B)+(2a-b)\frac{x\beta}{\alpha}[A,B]\right) + (2a^{m+1}-b^{m+1})C_m x^{m+1}+O(x^{n+2}). \quad (56)$$

To obtain an $(m+1)$st-order product formula, a, b should satisfy

$$2a-b=1, \quad 2a^{m+1}-b^{m+1}=0 \quad (57)$$

to eliminate the $(2a^{m+1}-b^{m+1})C_{m+1}x^{m+1}$ term. The solution is

$$a=(2-2^{1/(m+1)})^{-1}, \quad b=2^{1/(m+1)}a. \quad (58)$$

If m is odd, then we consider the two-copy sequence

$$f_{R,m}(-ax)^{-1}f_{R,m}(bx) = \exp\left((a+b)x(A+B)+(a+b)\frac{x\beta}{\alpha}[A,B]\right) + (-a^{m+1}+b^{m+1})C_m x^{m+1}+O(x^{n+2}). \quad (59)$$

To eliminate the $C_m x^{n+1}$ term, we must have

$$a+b=1, \quad -a^{m+1}+b^{m+1}=0, \quad (60)$$

which is satisfied with

$$a=b=\frac{1}{2}. \quad (61)$$

IV. NUMERICAL EVIDENCE

The analytical formulas presented above indicate how the errors scale with powers of x. However, the constant factors in the error terms of different product formulas significantly affect their performance in practice. To better understand this, we numerically compare the different approaches. Specifically, we evaluate the performance of our $\sqrt{10}$-copy, $\sqrt{6}$-copy, $\sqrt{5}$-copy, and $\sqrt{4}$-copy recursive formulas built from the base formula $S_3(x)$ and compare them with the previous best method, the Childs-Wiebe $\sqrt{6}$-copy formula [15] [which is built from the base formula $S_2(x)$].

We evaluate the commutator product formulas with $A=-i\sigma_x$ and $B=-i\sigma_z$ for $x\in[10^{-2},10^{-1}]$. Figure 3 plots the error compared with the exact exponential of the commutator $[A,B]$. The errors scale as $x^{6.001}$ ($\mathcal{G}_5$), $x^{5.958}$ (V_5), $x^{5.867}$ (W_5), $x^{6.371}$ (Q_5), and $x^{4.920}$ ($\widetilde{V}_4$), in good agreement with the analytical scalings [27].

Next, we compare the number of gates required to achieve a fixed accuracy for different recursive formulas. We set $e^{-i\sigma_x x}$, $e^{-i\sigma_z x}$ as our elementary exponentials and $\exp([-i\sigma_x x,-i\sigma_z x])$ as our target. We numerically determine the minimum number of elementary exponentials (i.e., gates) to achieve a fixed accuracy $\|f(x)-\exp([-i\sigma_x x,-i\sigma_z x])\|_2 = 10^{-4}$ for different approximation formulas $f(x)$.

We calculate the number of gates to achieve an error of at most 10^{-4} using the aforementioned product formulas. Specifically, we consider fifth-order product formulas obtained from the base formula S_3 using the $\sqrt{4}$-copy (Q_5), $\sqrt{5}$-copy (W_5), $\sqrt{6}$-copy (V_5), and $\sqrt{10}$-copy ($\mathcal{G}_5$) approaches. We compare them with the fourth-order formula $\widetilde{V}_4$ obtained from the base formula S_2 using the Childs-Wiebe $\sqrt{6}$-copy recursion [15]. See Fig. 4 for the numerical comparison. The number of gates for the $\sqrt{10}$-copy formula $\mathcal{G}_5$ is constant in the interval $x\in[0.1,0.3]$ since its error is always below the threshold. Asymptotically, the scaling of the number of gates for an nth-order formula to achieve fixed accuracy is $O(x^{\frac{2n+2}{n-1}})$. Figure 4 numerically shows that the $\sqrt{10}$-copy formula has the best performance. Although the $\sqrt{4}$-copy, $\sqrt{5}$-copy, $\sqrt{6}$-copy, and $\sqrt{10}$-copy formulas all have the same error

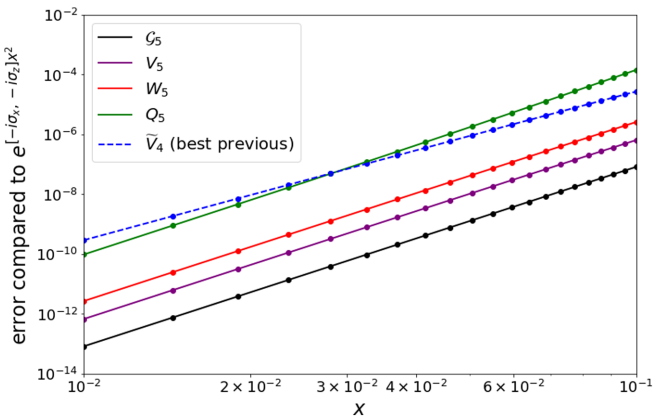

FIG. 3. Error scaling for $\mathcal{G}_5$, V_5, W_5, Q_5, and the previous best formula $\widetilde{V}_4$. We use the spectral norm $\|f(x)-\exp([-i\sigma_x,-i\sigma_z]x^2)\|$ to measure the error. By fitting the last ten points for each formula ($0.05\leqslant x\leqslant 0.1$), we find that the slopes for $\mathcal{G}_5$, V_5, W_5, Q_5, and $\widetilde{V}_4$ are 6.001, 5.958, 5.967, 6.371, and 4.920, respectively.

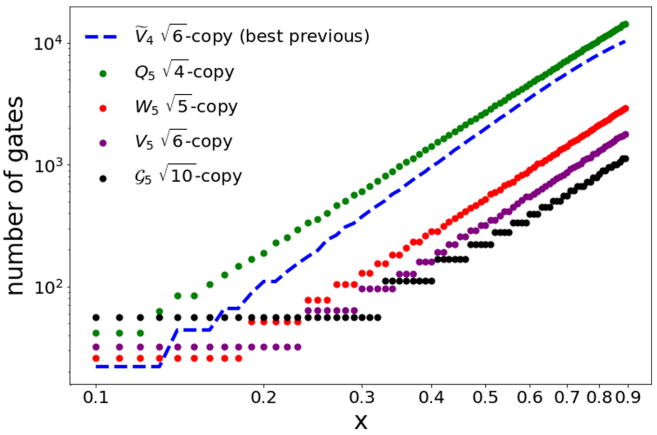

FIG. 4. Number of gates to achieve $\exp(x^2[-i\sigma_x, -i\sigma_z])$ within error 10^{-4}.

scaling, the constant factors determine their performance in practice.

Figure 5 shows the error in simulating $\exp([-i\sigma_x, -i\sigma_z])$ using $\widetilde{V}_4$ (the best previous method), Q_5, W_5, V_5, and $\mathcal{G}_5$. The horizontal axis indicates the total number of elementary exponentials, while the vertical axis indicates the simulation error $\|f(1/\sqrt{r})^r - \exp([-i\sigma_x, -i\sigma_z])\|_2$, where r is the number of time steps used in the simulation and $f(x)$ is the product formula. In an r-step simulation, the total number of elementary exponentials for Q_5, W_5, V_5, $\mathcal{G}_5$, and $\widetilde{V}_4$ is $21r$, $26r$, $32r$, $56r$, and $22r$, respectively. The numerical results show that the larger number of exponentials in each time step of V_5 and $\mathcal{G}_5$ is offset by their reduced error.

Figures 4 and 5 show that V_5 and $\mathcal{G}_5$ improve upon the best previous result, $\widetilde{V}_4$. Hybrid approaches that combine previous recursive formulas with the base formula $S_3(x)$ proposed in this paper also give improvements over $\widetilde{V}_4$; however, they do not perform as well as W_5, V_5, and $\mathcal{G}_5$, and so we do not include them in Figs. 4 and 5.

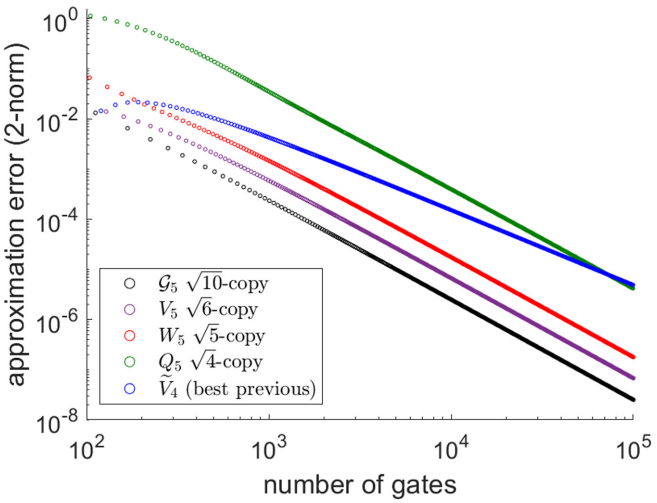

FIG. 5. Simulation error of $\exp([-i\sigma_x, -i\sigma_z])$ for different gate numbers and formulas.

V. APPLICATIONS TO QUANTUM SIMULATION

A. Counterdiabatic driving

In this section, we discuss using commutator product formulas to implement counterdiabatic driving (CD) [20–22,28]. In an adiabatic process $\mathcal{H}[\lambda(t)]$, the time evolution $\exp(-i\int dt\,\mathcal{H}[\lambda(t)])$ keeps the system in its instantaneous ground state if $\lambda(t)$ is slowing varying. In other words,

$$|\Psi(\tau)\rangle \approx \exp\left(-i\int_0^\tau dt\,\mathcal{H}[\lambda(t)]\right)|\Psi(0)\rangle, \tag{62}$$

where $|\Psi(t)\rangle$ denotes the ground state of $\mathcal{H}[\lambda(t)]$. In general, this approximation fails if $\lambda(t)$ varies too rapidly. However, by introducing counterdiabatic driving terms, the system can remain in the ground state even though $\lambda(t)$ varies rapidly. Specifically,

$$|\Psi(\tau)\rangle \approx \exp\left(-i\int_0^\tau dt\,\mathcal{H}_{\mathrm{CD}}[\lambda(t)]\right)|\Psi(0)\rangle, \tag{63}$$

where [23,28]

$$\mathcal{H}_{\mathrm{CD}}[\lambda(t)] = \mathcal{H}[\lambda(t)] + \dot{\lambda}C_\lambda, \tag{64}$$

with $\langle m|C_\lambda|n\rangle = -i\frac{\langle m|\partial_\lambda\mathcal{H}|n\rangle}{\epsilon_m-\epsilon_n}$, where $|n\rangle$ denotes an eigenstate of $\mathcal{H}(\lambda)$ with energy ϵ_n, i.e., $\mathcal{H}(\lambda)|n\rangle = \epsilon_n|n\rangle$.

Reference [23] proposes using Floquet engineering to generate the CD term C_λ. This term can be expressed as the sum of nested commutators [23]

$$C_\lambda = i\sum_k c_k(\lambda)\underbrace{[H,[H,\ldots[H}_{2k-1},\partial_\lambda H]], \tag{65}$$

where the coefficients $c_k(\lambda)$ are determined by minimizing the action

$$S_l = \mathrm{Tr}\left[G_l^2\right], \tag{66}$$

with

$$G_l = \partial_\lambda\mathcal{H} - i[\mathcal{H}, C_\lambda]. \tag{67}$$

For simplicity, we truncate to only the first term, giving

$$C_\lambda \approx ic_1(\lambda)[\mathcal{H}, \partial_\lambda\mathcal{H}]. \tag{68}$$

Commutator product formulas can be used to implement the CD term. To demonstrate this application, we consider the case $H(\lambda) = H_0 + \lambda H_1$. The time evolution is

$$\exp\left(-i\int dt\,\mathcal{H}_{\mathrm{CD}}(t)\right) = \exp\left(-\int dt\,(i\mathcal{H}_0 + i\lambda\mathcal{H}_1 - \dot{\lambda}c_1(\lambda)[\mathcal{H}_0,\mathcal{H}_1])\right). \tag{69}$$

For each infinitesimal time interval $[t, t+\delta t]$, we apply the following unitary operator:

$$\exp(-i\mathcal{H}_0\delta t - i\lambda\mathcal{H}_1\delta t + \dot{\lambda}c_1(\lambda)[\mathcal{H}_0,\mathcal{H}_1]\delta t). \tag{70}$$

This unitary operator can be simulated by the product formula equation (25) with $A = -i\mathcal{H}_1$, $B = -i\lambda\mathcal{H}_0$, $n = \frac{1}{\delta t}$, $\alpha = 1$, and $\beta = \frac{\dot{\lambda}c(\lambda)}{\lambda}$. The product formula error is $O((\beta+\beta^2)\delta t)$.

As a concrete example, consider using the product formula to simulate the counterdiabatic time evolution of the time-dependent two-qubit Hamiltonian $\mathcal{H}(\lambda) = \mathcal{H}_A + \mathcal{H}_B$, where

$$\mathcal{H}_A := h_z(\lambda - 1)\left(\sigma_1^z + \sigma_2^z\right),$$
$$\mathcal{H}_B := J\left(\sigma_1^x\sigma_2^x + \sigma_1^z\sigma_2^z\right), \tag{71}$$

with $\lambda(t) = \sin^2\left(\frac{\pi}{2}\sin^2\left(\frac{\pi t}{2\tau}\right)\right)$. Notice that $\mathcal{H}_A$ and $\mathcal{H}_B$ do not commute. The first-order counterdiabatic driving term is [23]

$$C_\lambda = -\frac{Jh_z}{2}\frac{\left(\sigma_1^y\sigma_2^x + \sigma_1^x\sigma_2^y\right)}{J^2 + 4(\lambda-1)^2h_z^2}. \tag{72}$$

We can write Eq. (72) as a commutator between $\mathcal{H}_A$ and $\mathcal{H}_B$:

$$C_\lambda = i\frac{1}{4(1-\lambda)}\frac{[-i\mathcal{H}_A, -i\mathcal{H}_B]}{J^2 + 4(\lambda-1)^2h_z^2}. \tag{73}$$

Hence we can construct the first-order counterdiabatic term using a commutator product formula. In the product formula setting, we choose $A = -i\mathcal{H}_A$, $B = -i\mathcal{H}_B$, $n = \frac{1}{\delta t}$, $\alpha = 1$, and $\beta = \dot{\lambda}\frac{1}{4(1-\lambda)}\frac{1}{J^2+4(\lambda-1)^2h_z^2}$ to implement the counterdiabatic Hamiltonian over a time interval δt. Defining $R = \frac{\beta n}{\alpha^2}$, the product formula equation (25) gives

$$\begin{aligned} f_R(\delta t) &= e^{p_1(R)A\delta t}e^{p_2(R)B\delta t}e^{p_3(R)A\delta t}\ e^{p_4(R)B\delta t}e^{p_5(R)A\delta t}e^{p_6(R)B\delta t} \\ &= \exp\left(\delta t(A+B) + \beta\delta t[A,B] + O(\delta t^2)\right) \\ &= \exp\left(-i(\mathcal{H}_A + \mathcal{H}_B)\delta t - i(\dot{\lambda}C_\lambda)\delta t\right) \\ &= \exp(-i\mathcal{H}_{\mathrm{CD}}\delta t), \end{aligned} \tag{74}$$

where $p_i(R)$ is the solution equation (19), which provides a good approximation provided that R is large (i.e., δt is small) [29]. The overall counterdiabatic time evolution is the product of $f_R(\delta t)$ for each time interval, i.e.,

$$\exp\left(-i\int_0^\tau dt\ \mathcal{H}_{\mathrm{CD}}[\lambda(t)]\right) = \prod_{k=0}^{N-1} f_{R(t_k)}(\delta t), \tag{75}$$

where $t_k = \frac{k}{N}\tau$ and $\delta t = \frac{\tau}{N}$. Notice that in each term, the operator A and the parameter R depend on t_k.

For comparison, the time evolution of the original Hamiltonian can be simulated as

$$\exp\left(-i\int_0^\tau dt\ \mathcal{H}[\lambda(t)]\right) = \prod_{k=0}^{N-1}(e^{-i\mathcal{H}_A(t_k)\delta t/3}e^{-i\mathcal{H}_B\delta t/3})^3, \tag{76}$$

where we use first-order Trotterization, $e^{A\delta t/3}e^{B\delta t/3} = e^{(A+B)\delta t/3} + O(\delta t^2)$. We choose $\delta t/3$ in each time step to match the total gate number in the counterdiabatic simulation, which uses six exponentials for each t_k.

We consider approximations to the evolution from $t = 0$ to $t' = r\delta t$ of $\mathcal{H}[\lambda(t)]$ and $\mathcal{H}_{\mathrm{CD}}[\lambda(t)]$ by standard Trotterization and our digital CD approach, respectively.

The evolved states with these approximations are

$$\begin{aligned} \left|\Psi_{\mathrm{Trotter}}^{\mathrm{evolved}}(t')\right\rangle &= \prod_{k=0}^{r-1}(e^{-i\mathcal{H}_A(t_k)\delta t/3}e^{-i\mathcal{H}_B\delta t/3})^3|\Psi(0)\rangle, \\ \left|\Psi_{\mathrm{CD}}^{\mathrm{evolved}}(t')\right\rangle &= \prod_{k=0}^{r-1} f_{R(t_k)}(\delta t)|\Psi(0)\rangle. \end{aligned} \tag{77}$$

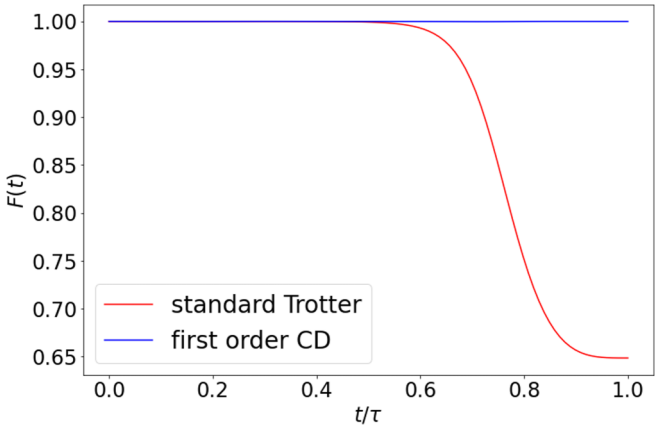

FIG. 6. The fidelity of the evolved state under standard Trotterization and CD protocols. The Hamiltonian has $J = -1$ and $h_z = 5$, the total evolution time is $\tau = 1$, and the number of steps is $N = 100$. There are six exponentials in each step, so each simulation uses 600 gates in total. The CD protocol remains close to the ground state, while the standard Trotterization approach starts to deviate from the ground state after $t \approx 0.6\tau$.

We define the fidelity of the process as the overlap with the ground state $|\Psi(t')\rangle$ of $\mathcal{H}[\lambda(t)]$:

$$F_\alpha(t') := \left|\langle\Psi(t')\left|\Psi_\alpha^{\mathrm{evolved}}(t')\right\rangle\right|^2, \quad \alpha \in \{\mathrm{Trotter}, \mathrm{CD}\}. \tag{78}$$

Figure 6 shows a numerical computation of these quantities. We see that the product formula simulation of counterdiabatic evolution uses the commutator term to keep the system close to the ground state, while the corresponding evolution with standard Trotterization deviates from the ground state after $t \approx 0.6\tau$.

We also examine the performance of the standard Trotter method and our digital CD approach for different numbers of gates in Fig. 7. When the number of gates is large, the digital CD protocol has higher fidelity than the standard Trotter protocol. The final fidelity is determined by the schedule $\lambda(t)$. Moreover, we see that even with fewer gates, the first-order CD approach has higher fidelity than the standard Trotter method.

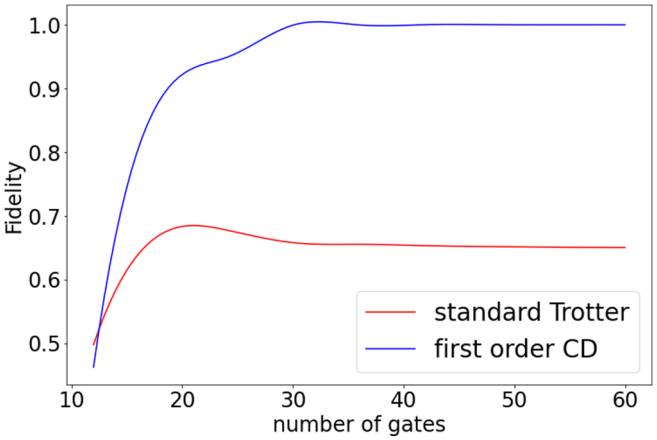

FIG. 7. Comparison of the final fidelity at $t = \tau$ using the standard Trotter protocol and counterdiabatic driving approach.

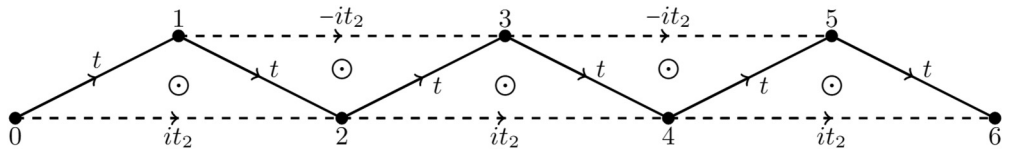

FIG. 8. A 1d fermion chain with both nearest-neighbor (NN) and next-nearest-neighbor (NNN) hopping. The NN hopping terms are all t, while the NNN terms are it_2 and $-it_2$ alternatively (t, t_2 are real). This corresponds to $\frac{\pi}{2}$ flux insertion in each triangle.

B. 1d fermion chain with next-nearest-neighbor hopping terms

In this section, we discuss how to generate the time evolution of a 1d fermion chain with nearest-neighbor (NN) and next-nearest-neighbor (NNN) hopping terms using only two-site gates acting on neighboring sites on a fermionic digital quantum simulator [30].

Consider a 1d fermion chain with one complex fermion on each site j. The fermion operators $c_j, c_j^\dagger$ satisfy the canonical anticommutation relations

$$\{c_j, c_k^\dagger\} = \delta_{jk}, \quad \{c_j, c_k\} = \{c_j^\dagger, c_k^\dagger\} = 0. \tag{79}$$

We partition the hopping terms into two sets: those between sites $2j$ and $2j+1$ and those between sites $2j+1$ and $2j+2$. More explicitly, we define

$$\begin{aligned} H_0 &:= (c_0^\dagger c_1 + c_2^\dagger c_3 + \cdots + c_{L-2}^\dagger c_{L-1}) + \text{H.c.}, \\ H_1 &:= (c_1^\dagger c_2 + c_3^\dagger c_4 + \cdots + c_{L-1}^\dagger c_0) + \text{H.c.}, \end{aligned} \tag{80}$$

where we use periodic boundary conditions and assume that the total number of sites L is even. Notice that the terms in H_0 pairwise commute, so the time evolution of H_0 is exactly the product of the time evolutions of the individual terms, and similarly for H_1. By choosing $A = iH_0$, $B = iH_1$, $\alpha = -tT$, and $\beta = -t_2 T$ in Eqs. (24) and (25), we have

$$f_{\frac{\beta n}{\alpha^2}}\left(\frac{\alpha}{n}\right) = \exp\left(-iH_{\text{eff}}\frac{T}{n} + O\left(\frac{LT^2}{n^2}\right)\right), \tag{81}$$

where

$$\begin{aligned} H_{\text{eff}} &= t(H_0 + H_1) + it_2[H_0, H_1] \\ &= t(c_0^\dagger c_1 + c_1^\dagger c_2 + c_2^\dagger c_3 + c_3^\dagger c_4 + \cdots + \text{H.c.}) \\ &\quad + t_2(ic_0^\dagger c_2 - ic_1^\dagger c_3 + ic_2^\dagger c_4 - ic_3^\dagger c_5 + \cdots + \text{H.c.}). \end{aligned} \tag{82}$$

Repeating the product formula n times, we find

$$f_{\frac{\beta n}{\alpha^2}}\left(\frac{\alpha}{n}\right)^n = \exp\left(-iH_{\text{eff}}T + O\left(\frac{LT^2}{n}\right)\right). \tag{83}$$

The effective Hamiltonian H_{eff} is shown in Fig. 8. It contains NN hopping terms with amplitude t and NNN hopping terms with amplitude t_2 and alternating i and $-i$ factors. Physically, this Hamiltonian corresponds to the insertion of a $\frac{\pi}{2}$ flux in each triangle. If we transform the NNN term $c_j^\dagger c_{j+2}$ to a qubit representation, it corresponds to a three-qubit interaction $\sigma_j^+ \sigma_{j+1}^z \sigma_{j+2}^-$ as occurs in lattice gauge theories [31,32].

To simulate H_0 or H_1, we use $\frac{L}{2}$ gates. To simulate the time evolution of H_{eff}, we use $6n \times \frac{L}{2} = 3nL$ gates, which is proportional to the number of steps n and the chain length L. Thus, using only nearest-neighbor terms (or two-qubit gates in the qubit representation), we are able to simulate the next-nearest-neighbor terms (or three-qubit gates) with the same number of gates as in standard Trotterization of a Hamiltonian with only nearest-neighbor terms [34]. We do not require any gate decomposition of three-qubit interactions.

C. Fractional quantum Hall phases on lattices

In Ref. [35], Kapit and Mueller showed that the addition of an appropriate next-nearest-neighbor hopping term to magnetic models on lattices can significantly flatten the lowest band. Such a band flattening can further stabilize and increase the gap of lattice quantum Hall states such as Laughlin states. In this section, we discuss how to simulate the Kapit-Mueller Hamiltonian with nearest-neighbor (NN) and next-nearest-neighbor (NNN) hopping terms on a two-dimensional square lattice using only the time evolution operators of nearest-neighbor hopping terms.

Using our product formula for both sum and commutator, as in Eq. (25), we can simulate the Hamiltonian of the Kapit-Mueller model. The general form of the Kapit-Mueller Hamiltonian can be regarded as a variation of the Hofstadter Hamiltonian [36], which involves not only NN hopping but also long-range hopping. The long-range hopping ensures a flat band, which can be regarded as a degenerate Landau level to stabilize fractional quantum Hall states. By truncating to only the NN and NNN terms, the Kapit-Mueller model simplifies the interactions while still demonstrating features of fractional quantum Hall phases.

Specifically, the Kapit-Mueller Hamiltonian is of the form

$$\begin{aligned} H_{\text{KM}} &= \sum_{\langle j,k\rangle, \langle\langle j,k\rangle\rangle} J(z_j, z_k) a_j^\dagger a_k, \quad \text{where} \\ J(z_j, z_k) &= K(z) e^{(\pi/2)(z_j z^* - z_j^* z)\phi}, \\ K(z) &= t \times (-1)^{x+y+xy} e^{-\frac{\pi}{2}[(1-\phi)|z|^2]}, \end{aligned} \tag{84}$$

with $z_j = x_j + iy_j$ denoting the position of the jth site, $z = z_k - z_j$ representing the displacement from the jth site to the kth site, $\langle j,k\rangle$ indicating that j,k are NN sites, and $\langle\langle j,k\rangle\rangle$ indicating that j,k are NNN sites. The $a_j^\dagger$ operator is a bosonic creation operator acting on site j, and ϕ is the magnetic flux inside each plaquette, as shown in Fig. 9. The hopping phase factor $(z_j z^* - z_j^* z)\phi$ corresponds to a vector potential in the symmetric gauge.

For convenience, we use Cartesian coordinates (m,n) to label lattice sites and rescale $\pi\phi$ to ϕ. The lattice constant is set to 1. We divide all the nearest-neighbor hopping terms into four parts:

$$H_1 = -J \sum_{m+n \text{ even}} e^{-in\phi} a_{m+1,n}^\dagger a_{m,n} + \text{H.c.},$$

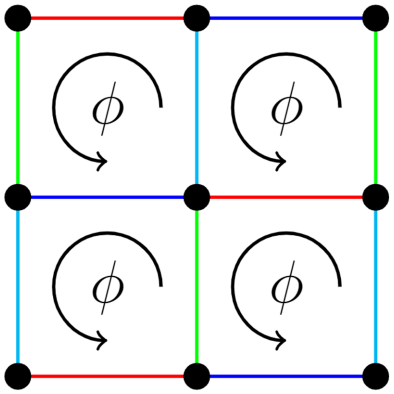

FIG. 9. Square lattice in a uniform magnetic field. The red links belong to H_1, the dark blue links belong to H_2, the green links belong to H_3, and the light blue links belong to H_4.

$$H_2 = -J \sum_{m+n \text{ odd}} e^{-in\phi} a^\dagger_{m+1,n} a_{m,n} + \text{H.c.},$$
$$H_3 = -J \sum_{m+n \text{ odd}} a^\dagger_{m,n+1} a_{m,n} + \text{H.c.},$$
$$H_4 = -J \sum_{m+n \text{ even}} a^\dagger_{m,n+1} a_{m,n} + \text{H.c.} \tag{85}$$

Using the bosonic commutation relation $[a_i, a_j^\dagger] = \delta_{i,j}$, we see that the commutator between two hopping terms that overlap on site j is

$$-i[K_1 a_i^\dagger a_j + \text{H.c.}, K_2 a_j^\dagger a_k + \text{H.c.}] = -iK_1 K_2 a_i^\dagger a_k + \text{H.c.} \tag{86}$$

Thus commutators between NN hopping terms can generate NNN hopping. We compute the following commutators between different NN hopping terms:

$$\begin{aligned}
-i[H_1, H_3] &= -J^2 \sum_{m+n=\text{odd}} ie^{-i(n+1)\phi} a^\dagger_{m+1,n+1} a_{m,n} + \text{H.c.} + J^2 \sum_{m+n=\text{even}} ie^{-in\phi} a^\dagger_{m+1,n+1} a_{m,n} + \text{H.c.},\\
-i[H_1, H_4] &= -J^2 \sum_{m+n=\text{even}} ie^{-in\phi} a^\dagger_{m+1,n} a_{m,n+1} + \text{H.c.} + J^2 \sum_{m+n=\text{odd}} ie^{-i(n+1)\phi} a^\dagger_{m+1,n} a_{m,n+1} + \text{H.c.},\\
-i[H_2, H_3] &= -J^2 \sum_{m+n=\text{odd}} ie^{-in\phi} a^\dagger_{m+1,n} a_{m,n+1} + \text{H.c.} + J^2 \sum_{m+n=\text{even}} ie^{-i(n+1)\phi} a^\dagger_{m+1,n} a_{m,n+1} + \text{H.c.},\\
-i[H_2, H_4] &= -J^2 \sum_{m+n=\text{even}} ie^{-i(n+1)\phi} a^\dagger_{m+1,n+1} a_{m,n} + \text{H.c.} + J^2 \sum_{m+n=\text{odd}} ie^{-in\phi} a^\dagger_{m+1,n+1} a_{m,n} + \text{H.c.}
\end{aligned} \tag{87}$$

Therefore

$$\begin{aligned}
-i[H_1, H_3] - i[H_2, H_4] + i[H_1, H_4] + i[H_2, H_3] &= -2J^2 \sin\left(\frac{\phi}{2}\right) \sum_{m,n} e^{-i(n+\frac{1}{2})\phi} (a^\dagger_{m+1,n+1} a_{m,n} + a^\dagger_{m+1,n} a_{m,n+1}) + \text{H.c.}\\
&= -i[H_1 - H_2, H_3 - H_4].
\end{aligned} \tag{88}$$

Comparing the phase factors of NN hopping in Eq. (85) and NNN hopping in Eq. (87), we find the effective Hamiltonian in terms of H_1, H_2, H_3, H_4 and their commutators,

$$H_{\text{eff}} = H_1 + H_2 + H_3 + H_4 - iJ'[H_1 - H_2, H_3 - H_4], \tag{89}$$

which describes a system with NN and NNN hopping where the magnetic flux inside each plaquette is $\phi/2$. Our product formula naturally generates the phase factor corresponding to a uniform magnetic field applied to the lattice. Since we can control the coefficient of the commutator term in the product formula, H_{eff} is equivalent to the Kapit-Mueller Hamiltonian with the choice $J' = \frac{\exp(\phi/4 - \pi/2)}{2J \sin(\phi/2)}$, which is fine-tuned to realize a flat band. We can use our product formula equation (28) to simulate H_{eff} by setting $A = i(H_1 - H_2)$, $B = i(H_3 - H_4)$, and $C = i(2H_2 + 2H_4)$ and taking $\alpha = 1$, $\beta = J'$.

VI. DISCUSSION

We conclude by discussing some possible future research directions.

Trajectories of product formulas.. Given a product formula

$$e^{p_1 xA} e^{p_2 xB} e^{p_3 xA} e^{p_4 xB} e^{p_5 xA} e^{p_6 xB} \cdots, \tag{90}$$

we can plot the "time evolution trajectory" of the coefficients of A and B, namely, $(p_1, p_1 + p_3, p_1 + p_3 + p_5, \ldots)$ and $(p_2, p_2 + p_4, p_2 + p_4 + p_6, \ldots)$, respectively. Reference [13] studies product formulas for the sum $A + B$, with the time evolution trajectory starting from $t = 0$ and ending at $t = 1$. The authors suggest that a good product formula for the sum should have the entire time evolution trajectory inside the "allowed" interval [0,1], since times outside this interval do not correspond to the evolution under consideration.

For commutators, the time evolution trajectory starts and ends at $t = 0$, so we do not have an "allowed" interval. However, we can still plot the time evolution trajectories, as shown in Fig. 10 for the $\sqrt{4}$-copy and $\sqrt{10}$-copy product formulas. We see that the $\sqrt{10}$-copy product formula has a smaller range for the time evolution trajectories of A and B. This may explain why the $\sqrt{10}$-copy product formula performs better than the $\sqrt{4}$-copy product formula. Similar considerations hold for other formulas (for example, the ranges of the trajectories for the $\sqrt{5}$-copy formula are intermediate between those of the $\sqrt{4}$- and $\sqrt{10}$-copy formulas). It might be useful to develop a more general and quantitative understanding of how the trajectories of a product formula affect its performance.

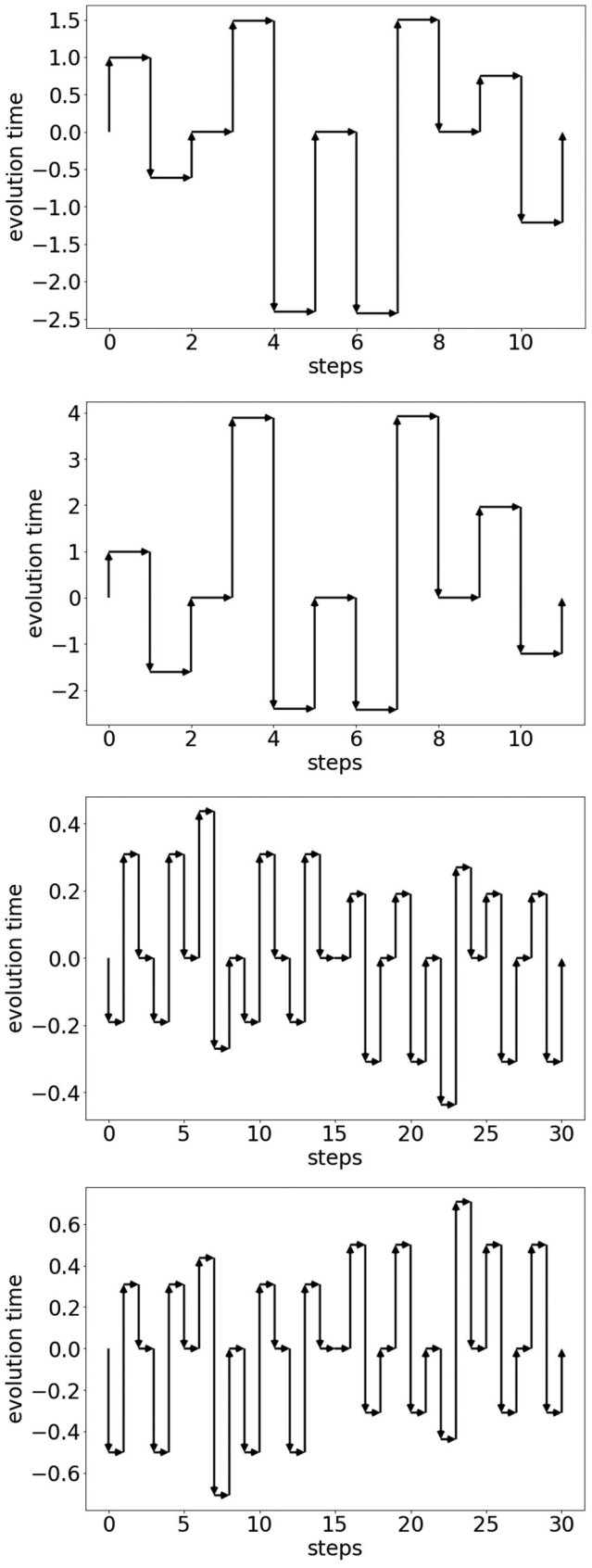

FIG. 10. Time evolution trajectories of product formulas. From top to bottom, $Q_5(1)$ ($\sqrt{4}$ copy) for A and B and $\mathcal{G}_5(1)$ ($\sqrt{10}$ copy) for A and B.

Optimal recursive relation. We argue that the $\sqrt{4}$-copy recursive relation may use the smallest possible number of copies to generate higher-order product formulas. In other words, it seems unlikely that a recursive relation could use fewer than two copies of an nth-order product formula to generate an $(n+1)$st-order product formula. To obtain a pth-order product formula, the coefficients of pth-order commutators must be eliminated. If p is a prime, the number of independent commutators is $\frac{2^p-2}{p}$, which is the dimension of the graded component of a free Lie algebra with length p [18,19,37]. A $(2-\epsilon)$-copy recursive formula would use a number of exponentials proportional to $(2-\epsilon)^p \ll \frac{2^p-2}{p}$, which would mean using far fewer parameters than the number of polynomial equations to be solved. Thus we conjecture that there is no $(2-\epsilon)$-copy recursive formula.

Direct derivation of a fourth-order commutator product formula. To better understand possible direct constructions of commutator product formulas, we would like to solve the polynomial equations for the fourth-order case, namely [38],

$$e^{p_1 xA} e^{p_2 xB} e^{p_3 xA} e^{p_4 xB} e^{p_5 xA} e^{p_6 xB} \cdots = e^{x^2[A,B]} + O(x^5). \quad (91)$$

We first revisit the polynomial equations (16) for the third-order product formula. We can rewrite these equations as

$$\begin{aligned} &\mathcal{A} = 0, \qquad &&\mathcal{B} = 0, \qquad &&\mathcal{BA} = -1, \\ &\mathcal{ABA} = 0, \qquad &&\mathcal{BAB} = 0, \end{aligned} \quad (92)$$

where

$$\mathcal{A} := \sum_{i \text{ odd}} p_i = p_1 + p_3 + p_5 + \cdots \quad (93)$$

represents the sum of all coefficients of the A term and

$$\mathcal{B} := \sum_{i \text{ even}} p_i = p_2 + p_4 + p_6 + \cdots \quad (94)$$

represents the sum of all coefficients of the B term. Then

$$\mathcal{BA} := \sum_{i \text{ even},\, j \text{ odd},\, i<j} p_i p_j \quad (95)$$

is the sum of all coefficients of the BA term, and similarly

$$\begin{aligned} \mathcal{ABA} &:= \sum_{i \text{ odd},\, j \text{ even},\, k \text{ odd},\, i<j<k} p_i p_j p_k, \\ \mathcal{BAB} &:= \sum_{i \text{ even},\, j \text{ odd},\, k \text{ even},\, i<j<k} p_i p_j p_k. \end{aligned} \quad (96)$$

Following the same strategy, we can derive polynomial equations for the fourth-order commutator product formula:

$$\begin{aligned} &\mathcal{A} = 0, \qquad &&\mathcal{B} = 0, \\ &\mathcal{BA} = -1, \qquad &&\mathcal{ABA} = 0, \\ &\mathcal{BAB} = 0, \qquad &&\mathcal{A}^2\mathcal{BA} = 0, \\ &\mathcal{B}^2\mathcal{AB} = 0, \qquad &&\mathcal{ABAB} - \mathcal{BABA} = 0, \end{aligned} \quad (97)$$

where we have defined

$$\mathcal{A}^2\mathcal{BA} := \sum_{i \text{ odd},\, j \text{ even},\, k \text{ odd},\, i<j<k} \frac{1}{2} p_i^2 p_j p_k$$

$$+\sum_{\substack{i\text{ odd},\ j\text{ odd},\ k\text{ even},\\ l\text{ odd},\ i<j<k<l}} p_i p_j p_k p_l,$$

$$\mathcal{B}^2\mathcal{A}\mathcal{B} := \sum_{i\text{ even},\ j\text{ odd},\ k\text{ even},\ i<j<k} \frac{1}{2}p_i^2 p_j p_k + \sum_{\substack{i\text{ even},\ j\text{ even},\ k\text{ odd},\\ l\text{ even},\ i<j<k<l}} p_i p_j p_k p_l,$$

$$\mathcal{A}\mathcal{B}\mathcal{A}\mathcal{B} := \sum_{\substack{i\text{ odd},\ j\text{ even},\ k\text{ odd},\\ l\text{ even},\ i<j<k<l}} p_i p_j p_k p_l,$$

$$\mathcal{B}\mathcal{A}\mathcal{B}\mathcal{A} := \sum_{\substack{i\text{ even},\ j\text{ odd},\ k\text{ even},\\ l\text{ odd},\ i<j<k<l}} p_i p_j p_k p_l. \tag{98}$$

However, we do not have an analytical solution for these polynomial equations.

Applications of counterdiabatic driving. In Sec. V A, we gave a two-qubit example to demonstrate the potential effectiveness of counterdiabatic driving in digital quantum computers. We believe this approach can be used to prepare ground states of spin-chain systems with high fidelity. Looking beyond quantum simulation, there might exist other efficient quantum algorithms based on counterdiabatic driving.

ACKNOWLEDGMENTS

We thank Mathias Van Regemortel, Bowen Yang, and Yixu Wang for helpful discussions. Y.-A.C. is supported by the JQI fellowship. A.M.C. received support from the National Science Foundation (Grant No. CCF-1813814 and QLCI Grant No. OMA-2120757) and the Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research in Quantum Computing program. Y.X., H.K., and M.H. were supported by ARO Grant No. W911NF-15-1-0397, National Science Foundation QLCI Grant No. OMA-2120757, AFOSR-MURI Grant No. FA9550-19-1-0399, the Department of Energy QSA program, and the Simons Foundation.

APPENDIX A: THE OPERATOR DIFFERENTIAL METHOD AND COMMUTATOR PRODUCT FORMULAS

In this Appendix, we introduce the operator differential method and show how it can be used to derive product formulas. We have [13]

$$e^{p_1 xA}e^{p_2 xB}e^{p_3 xA}e^{p_4 xB}e^{p_5 xA}e^{p_6 xB} = e^{\Phi(x)}, \tag{A1}$$

with

$$\Phi(x) = \sum_{k=0}^{\infty}\frac{(-1)^k}{k+1}\int_0^x (e^{p_1 t\delta_A}e^{p_2 t\delta_B}e^{p_3 t\delta_A}e^{p_4 t\delta_B}e^{p_5 t\delta_A}e^{p_6 t\delta_B}-1)^k \times (p_1 A + e^{p_1 t\delta_A}p_2 B + e^{p_1 t\delta_A}e^{p_2 t\delta_B}p_3 A\cdots)\,dt, \tag{A2}$$

where $\delta_A O := [A, O]$ is called the operator differential. For example, the x term in $\Phi(x)$ comes from integrating the constant term in Eq. (A2):

$$\int_0^x (p_1 A + p_2 B + p_3 A + p_4 B + p_5 A + p_6 B)dt = (lA + mB)x, \tag{A3}$$

where

$$l := p_1 + p_3 + p_5,$$
$$m := p_2 + p_4 + p_6. \tag{A4}$$

Notice that only the $k = 0$ part contributes to the constant term in the integrand. The x^2 term in $\Phi(x)$ comes from the t terms in the integrand, which have two contributions: From the $k = 0$ part, we have

$$\int_0^x dt\, t[p_1(p_2+p_4+p_6)+p_3(p_4+p_6)+p_5p_6]\delta_A B + t[p_2(p_3+p_5)+p_4p_5]\delta_B A = \frac{x^2}{2}[p_1(p_2+p_4+p_6)+p_3(p_4+p_6)+p_5p_6 - p_2(p_3+p_5)-p_4p_5]\delta_A B, \tag{A5}$$

and from the $k = 1$ part, we have

$$-\frac{1}{2}\int_0^x dt\, t(p_2+p_4+p_6)(p_1+p_3+p_5)\delta_B A + t(p_1+p_3+p_5)(p_2+p_4+p_6)\delta_A B = 0. \tag{A6}$$

The x^2 term is therefore

$$\frac{x^2}{2}(lm-2q)\delta_A B, \tag{A7}$$

where

$$q := p_2p_3 + p_2p_5 + p_4p_5. \tag{A8}$$

The x^3 term in $\Phi(x)$ has three contributions. The first comes from the $k = 0$ part:

$$\int_0^x dt\frac{t^2}{2}\big[p_1^2(p_2+p_4+p_6)+p_3^2(p_4+p_6)+p_5^2p_6 + 2p_1p_3(p_4+p_6)+2(p_1+p_3)p_5p_6 - 2p_1p_2(p_3+p_5)-2(p_1+p_3)p_4p_5\big]\delta_A^2 B + \frac{t^2}{2}\big[p_2^2(p_3+p_5)+p_4^2p_5+2p_2p_4p_5 - 2p_2p_3(p_4+p_6)-2(p_2+p_4)p_5p_6\big]\delta_B^2 A. \tag{A9}$$

Defining

$$r := p_1p_2p_3 + p_1p_2p_5 + p_1p_4p_5 + p_3p_4p_5,$$
$$s := p_2p_3p_4 + p_2p_3p_6 + p_2p_5p_6 + p_4p_5p_6, \tag{A10}$$

the $k = 0$ part can be simplified as

$$\int_0^x dt\frac{t^2}{2}\big[(l(lm-q)-3r)\delta_A^2 B + (mq-3s)\delta_B^2 A\big] = \frac{x^3}{6}\big[(l(lm-q)-3r)\delta_A^2 B + (mq-3s)\delta_B^2 A\big]. \tag{A11}$$

Similarly, for $k = 1$ we have

$$-\frac{1}{2}\int_0^x dt\, t(l\delta_A + m\delta_B)t[(lm-2q)\delta_A B] + \frac{1}{2}t^2\big(l^2\delta_A^2 + m^2\delta_B^2 + 2(lm-q)\delta_A\delta_B + 2q\delta_B\delta_A\big) \times (lA+mB) = -\frac{x^3}{6}\left[\left(\frac{1}{2}l^2m - lq\right)\delta_A^2 B - \left(\frac{1}{2}m^2 l - mq\right)\delta_B^2 A\right], \quad \text{(A12)}$$

and for $k = 2$ we have

$$\frac{1}{3}\int_0^x dt\, t^2(l\delta_A + m\delta_B)^2(lA+mB) = \frac{1}{3}\int_0^x t^2\delta_{lA+mB}^2(lA+mB) = 0. \quad \text{(A13)}$$

Overall, the x^3 term is

$$\frac{x^3}{6}\left[\left(\frac{l^2m}{2} - 3r\right)\delta_A^2 B + \left(\frac{m^2 l}{2} - 3s\right)\delta_B^2 A\right]. \quad \text{(A14)}$$

1. Pure commutators

To give a pure commutator formula, we would like to find $(p_1, p_2, p_3, p_4, p_5, p_6)$ such that

$$\Phi(x) = R[A,B]x^2 + O(x^4) \quad \text{(A15)}$$

for some constant R. For the first-order term to vanish, we require

$$l = p_1 + p_3 + p_5 = 0, \quad m = p_2 + p_4 + p_6 = 0. \quad \text{(A16)}$$

The x^2 term, Eq. (A7), and x^3 term, Eq. (A14), contribute

$$-x^2 q\delta_A B - \frac{x^3}{2}\big(r\delta_A^2 B + s\delta_B^2 A\big). \quad \text{(A17)}$$

To eliminate the third-order term, we need to solve

$$\begin{aligned} l &= p_1 + p_3 + p_5 = 0,\\ m &= p_2 + p_4 + p_6 = 0,\\ q &= p_2p_3 + p_2p_5 + p_4p_5 = -1,\\ r &= p_1p_2p_3 + p_1p_2p_5 + p_1p_4p_5 + p_3p_4p_5 = 0,\\ s &= p_2p_3p_4 + p_2p_3p_6 + p_2p_5p_6 + p_4p_5p_6 = 0. \end{aligned} \quad \text{(A18)}$$

One can check that the following choice satisfies the equations:

$$\begin{aligned} p_1 &= \frac{\sqrt{5}-1}{2}, \quad p_2 = \frac{\sqrt{5}-1}{2}, \quad p_3 = -1,\\ p_4 &= -\frac{\sqrt{5}+1}{2}, \quad p_5 = \frac{3-\sqrt{5}}{2}, \quad p_6 = 1. \end{aligned} \quad \text{(A19)}$$

Thus we have the explicit product formula equation (14).

2. Sums and commutators

Now consider a case where we include both linear and commutator terms, namely,

$$\Phi(x) = (A+B)x + R[A,B]x^2 + O(x^4), \quad \text{(A20)}$$

for an arbitrary constant R. The first-order term requires

$$l = p_1 + p_3 + p_5 = 1, \quad m = p_2 + p_4 + p_6 = 1, \quad \text{(A21)}$$

and the x^2 and x^3 terms are

$$\frac{x^2}{2}(1-2q) + \frac{x^3}{6}\left[\left(\frac{1}{2} - 3r\right)\delta_A^2 B + \left(\frac{1}{2} - 3s\right)\delta_B^2 A\right], \quad \text{(A22)}$$

which agrees with the derivation in Ref. [13]. Therefore the equations to be solved are

$$\begin{aligned} l &= p_1 + p_3 + p_5 = 1,\\ m &= p_2 + p_4 + p_6 = 1,\\ q &= p_2p_3 + p_2p_5 + p_4p_5 = -R + \tfrac{1}{2},\\ r &= p_1p_2p_3 + p_1p_2p_5 + p_1p_4p_5 + p_3p_4p_5 = \tfrac{1}{6},\\ s &= p_2p_3p_4 + p_2p_3p_6 + p_2p_5p_6 + p_4p_5p_6 = \tfrac{1}{6}. \end{aligned} \quad \text{(A23)}$$

APPENDIX B: EXISTENCE OF THE $\sqrt{4}$-COPY RECURSIVE FORMULA

In this Appendix, we prove that a real solution of

$$\begin{aligned} a^{n+1} - b^{n+1} + c^{n+1} - d^{n+1} &= 0,\\ a^{n+2} - b^{n+2} + c^{n+2} - d^{n+2} &= 0,\\ a^2 - b^2 + c^2 - d^2 &\neq 0 \end{aligned} \quad \text{(B1)}$$

exists for any odd $n = 2k-1$. We first take $a = 1$ and $b = 2$, giving

$$c^{2k} - d^{2k} = 2^{2k} - 1, \quad \text{(B2)}$$

$$c^{2k+1} - d^{2k+1} = 2^{2k+1} - 1. \quad \text{(B3)}$$

The solution $(c,d) = (2,1)$ is trivial since $a^2 - b^2 + c^2 - d^2 = 0$. We show that there exists a solution of the form $(c,d) = (2-\epsilon_2, -1+\epsilon_1)$ with $1 \geqslant \epsilon_1, \epsilon_2 \geqslant 0$.

For any positive integer k, the solutions of Eqs. (B2) and (B3) line on curves in the (ϵ_1, ϵ_2) plane. We can check that the solutions of Eq. (B2) include the two points $(\epsilon_1, \epsilon_2) = (0,0)$, $(1, y_1)$ with $y_1 := 2 - (2^{2k} - 1)^{\frac{1}{2k}}$. Similarly, the solutions of Eq. (B3) include $(\epsilon_1, \epsilon_2) = (0, y_2)$ and $(1, y_3)$ with $y_2 = 2 - (2^{2k+1} - 2)^{\frac{1}{2k+1}}$ and $y_3 = 2 - (2^{2k+1} - 1)^{\frac{1}{2k+1}}$. Notice that $y_1 > y_2 > y_3$. As shown in Fig. 11, the curve for Eq. (B2) is monotonically increasing, and the curve for Eq. (B3) is monotonically decreasing, so these curves must intersect in the region $0 < \epsilon_1 < 1$, providing a simultaneous solution of the two equations.

The final step is to show $a^2 - b^2 + c^2 - d^2 \neq 0$, or equivalently,

$$(2-\epsilon_2)^2 - (1-\epsilon_1)^2 \neq 3. \quad \text{(B4)}$$

It suffices to show that

$$(2-\epsilon_2)^2 - (1-\epsilon_1)^2 = 3 \quad \text{(B5)}$$

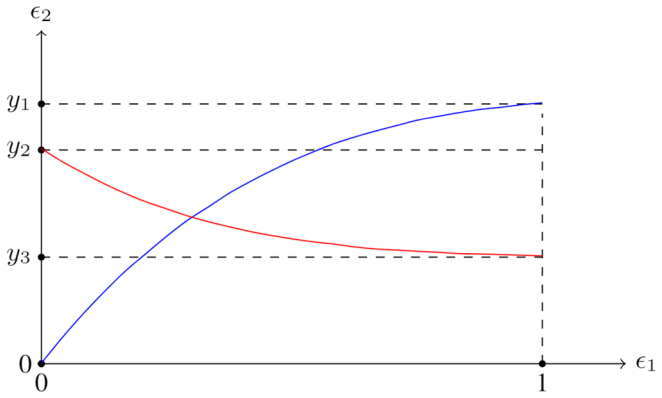

FIG. 11. The blue curve is the solution of Eq. (B2), a monotonically increasing function between (0,0) and $(1, y_1)$. The red curve is the solution of (B3), a monotonically decreasing function between $(0, y_2)$ and $(1, y_3)$. They intersect in the region $0 < \epsilon_1 < 1$. The plot shows the case $k = 2$, but the curves are qualitatively similar for any k.

and

$$(2-\epsilon_2)^{2k} - (1-\epsilon_1)^{2k} = 2^{2k} - 1 \tag{B6}$$

have no intersection in the interval $0 < \epsilon_1 < 1$ for $k > 1$. For each equation, we can think of ϵ_2 as a function of ϵ_1, i.e.,

$$f_1(x) := 2 - [3 + (1-x)^2]^{\frac{1}{2}},$$
$$f_2(x) := 2 - [2^{2k} - 1 + (1-x)^{2k}]^{\frac{1}{2k}}, \tag{B7}$$

corresponding to Eqs. (B5) and (B6), respectively. Now consider their derivatives:

$$f_1'(x) = \frac{1-x}{[3 + (1-x)^2]^{\frac{1}{2}}},$$
$$f_2'(x) = \frac{(1-x)^{2k-1}}{[2^{2k} - 1 + (1-x)^{2k}]^{\frac{2k-1}{2k}}}. \tag{B8}$$

For $0 < x < 1$ and $k > 1$, we have $(1-x) > (1-x)^{2k-1}$ and $[3 + (1-x)^2]^{\frac{1}{2}} < [2^{2k} - 1 + (1-x)^{2k}]^{\frac{2k-1}{2k}}$, so $f_1'(x) > f_2'(x)$. Starting from the initial point $f_1(0) = f_2(0) = 0$, $f_1(x)$ and $f_2(x)$ cannot intersect in the interval $0 < x < 1$. Therefore $a^2 - b^2 + c^2 - d^2 \neq 0$, and the solution for Eq. (51) always exists.

[1] R. P. Feynman, Simulating physics with computers, Int. J. Theor. Phys. **21**, 467 (1982).

[2] S. Lloyd, Universal quantum simulators, Science **273**, 1073 (1996).

[3] A. M. Childs and Y. Su, Nearly Optimal Lattice Simulation by Product Formulas, Phys. Rev. Lett. **123**, 050503 (2019).

[4] M. C. Tran, S.-K. Chu, Y. Su, A. M. Childs, and A. V. Gorshkov, Destructive Error Interference in Product-Formula Lattice Simulation, Phys. Rev. Lett. **124**, 220502 (2020).

[5] A. M. Childs, D. Maslov, Y. Nam, N. J. Ross, and Y. Su, Toward the first quantum simulation with quantum speedup, Proc. Natl. Acad. Sci. USA **115**, 9456 (2018).

[6] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu, Theory of Trotter Error with Commutator Scaling, Phys. Rev. X **11**, 011020 (2021).

[7] A. F. Shaw, P. Lougovski, J. R. Stryker, and N. Wiebe, Quantum algorithms for simulating the lattice Schwinger model, Quantum **4**, 306 (2020).

[8] R. Babbush, J. McClean, D. Wecker, A. Aspuru-Guzik, and N. Wiebe, Chemical basis of Trotter-Suzuki errors in quantum chemistry simulation, Phys. Rev. A **91**, 022311 (2015).

[9] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, Elucidating reaction mechanisms on quantum computers, Proc. Natl. Acad. Sci. USA **114**, 7555 (2017).

[10] H. De Raedt and B. De Raedt, Applications of the generalized Trotter formula, Phys. Rev. A **28**, 3575 (1983).

[11] K. E. Schmidt and M. A. Lee, High-accuracy Trotter-formula method for path integrals, Phys. Rev. E **51**, 5495 (1995).

[12] W. Janke and T. Sauer, Properties of higher-order Trotter formulas, Phys. Lett. A **165**, 199 (1992).

[13] N. Hatano and M. Suzuki, Finding exponential product formulas of higher orders, in *Quantum Annealing and Other Optimization Methods* (Springer, Berlin, 2005), pp. 37–68.

[14] F. Jean and P.-V. Koseleff, Elementary approximation of exponentials of Lie polynomials, in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, edited by T. Mora and H. Mattson (Springer, Berlin, 1997), pp. 174–188.

[15] A. M. Childs and N. Wiebe, Product formulas for exponentials of commutators, J. Math. Phys. (Melville, NY) **54**, 062202 (2013).

[16] When we multiply adjacent terms f_m and f_m^{-1} in the recursive formula, two gates in the middle may combine into a single gate if they are both exponentials of the same operator. In the best case, the number of gates is $N_{m+1} = pN_m - (p-1)$.

[17] The number of linearly independent commutators involving p copies of A or B is $\frac{2^p-2}{p}$ if p is a prime [18,19]. Canceling this many terms generically requires a number of parameters that is exponential in p.

[18] C. Reutenauer, *Free Lie Algebras* (Oxford University Press, New York, 1993).

[19] N. Bourbaki, *Lie Groups and Lie Algebras* (Springer, New York, 2008).

[20] M. Demirplak and S. A. Rice, Adiabatic population transfer with control fields, J. Phys. Chem. A **107**, 9937 (2003).

[21] M. Demirplak and S. A. Rice, Assisted adiabatic passage revisited, J. Phys. Chem. B **109**, 6838 (2005).

[22] D. Sels and A. Polkovnikov, Minimizing irreversible losses in quantum systems by local counterdiabatic driving, Proc. Natl. Acad. Sci. USA **114**, E3909 (2017).

[23] P. W. Claeys, M. Pandey, D. Sels, and A. Polkovnikov, Floquet-Engineering Counterdiabatic Protocols in Quantum Many-Body Systems, Phys. Rev. Lett. **123**, 090602 (2019).

[24] There are five equations in (16) and six variables $p_1, p_2, \ldots, p_6$. In principle, infinitely many solutions exist. We choose one with $p_6 = 1$ since it produces a concise formula involving the golden

ratio. Different solutions will perform differently, but in practice these differences appear to be small.

[25] M. Suzuki, Fractal decomposition of exponential operators with applications to many-body theories and Monte Carlo simulations, Phys. Lett. A **146**, 319 (1990).

[26] M. Suzuki, General theory of fractal path integrals with applications to many-body theories and statistical physics, J. Math. Phys. (Melville, NY) **32**, 400 (1991).

[27] We present this simple example in which A and B are 2×2 Pauli matrices to demonstrate the performance of product formulas, but we expect similar conclusions to hold more generally. We have also considered large random matrices A and B (up to size 50×50), and the relative performance of the product formulas is qualitatively similar.

[28] M. V. Berry, Transitionless quantum driving, J. Phys. A: Math. Theor. **42**, 365303 (2009).

[29] For arbitrary R, the solution $p_i(R)$ can be determined numerically using Eq. (18). However, here we choose δt to be small so that the solution $p_i(R)$ has the convenient form of Eq. (19).

[30] The following derivation uses fermionic operators $c, c^\dagger$, which can be transformed to Pauli operators by the Jordan-Wigner transformation. Thus this model can be realized by quantum computers with qubits using only two-qubit interactions.

[31] C. Schweizer, F. Grusdt, M. Berngruber, L. Barbiero, E. Demler, N. Goldman, I. Bloch, and M. Aidelsburger, Floquet approach to $\mathbb{Z}_2$ lattice gauge theories with ultracold atoms in optical lattices, Nat. Phys. **15**, 1168 (2019).

[32] In certain systems, there may be other ways of efficiently generating such a three-qubit term. For example, in the Google quantum computer, this term can be realized using a few single-qubit and imaginary SWAP (iSWAP) gates [33]. In general, the best circuit depends on the details of the physical platform. The preceding discussion provides an alternative simulation method, which may benefit some quantum simulators.

[33] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, A. Bengtsson, S. Boixo, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, Y.-A. Chen, B. Chiaro, R. Collins, S. J. Cotton, W. Courtney *et al.*, Observation of separated dynamics of charge and spin in the Fermi-Hubbard model, arXiv:2010.07965.

[34] In the standard Trotter approach, the evolution time for each step is the same, and the effective Hamiltonian is the sum of the simulated terms. In the algorithm discussed here, the evolution times are fine-tuned such that the next-nearest-neighbor terms appear in the effective Hamiltonian.

[35] E. Kapit and E. Mueller, Exact Parent Hamiltonian for the Quantum Hall States in a Lattice, Phys. Rev. Lett. **105**, 215303 (2010).

[36] D. R. Hofstadter, Energy levels and wave functions of Bloch electrons in rational and irrational magnetic fields, Phys. Rev. B **14**, 2239 (1976).

[37] The length is the number of times A and B occur in Lie brackets. For example, there are two terms of length 3: $[A, [A, B]]$ and $[B, [B, A]]$.

[38] While the number of terms in the product formula is finite, we do not indicate where it ends since we do not know *a priori* how many terms are required.