

# Simulating Quantum Dynamics On A Quantum Computer

Nathan Wiebe,<sup>1,2</sup> Dominic W. Berry,<sup>2</sup> Peter Høyer,<sup>1,3</sup> and Barry C. Sanders<sup>1,4</sup>

<sup>1</sup>*Institute for Quantum Information Science, University of Calgary, Alberta T2N 1N4, Canada*

<sup>2</sup>*Institute for Quantum Computing, University of Waterloo, Ontario N2L 3G1, Canada*

<sup>3</sup>*Department of Computer Science, University of Calgary, Alberta T2N 1N4, Canada*

<sup>4</sup>*Department of Physics & Astronomy, University of Calgary, Alberta T2N 1N4, Canada*

We present efficient quantum algorithms for simulating time-dependent Hamiltonian evolution of general input states using an oracular model of a quantum computer. Our algorithms use either constant or adaptively chosen time steps and are significant because they are the first to have time-complexities that are comparable to the best known methods for simulating time-independent Hamiltonian evolution, given appropriate smoothness criteria on the Hamiltonian are satisfied. We provide a thorough cost analysis of these algorithms that considers discretization errors in both the time and the representation of the Hamiltonian. In addition, we provide the first upper bounds for the error in Lie-Trotter-Suzuki approximations to unitary evolution operators, that use adaptively chosen time steps.

## I. INTRODUCTION

The original motivation for quantum computers stemmed from Feynman's famous conjecture that quantum computers could efficiently simulate quantum physical systems [1], whereas there is no known way to achieve this with classical computers. This conjecture has spurred the construction of a number of quantum algorithms to efficiently simulate quantum systems under a Hamiltonian [2–9]. However, these algorithms are primarily for time-independent Hamiltonians. A simple extension to time-dependent Hamiltonians yields complexity scaling quadratically with the simulation time [10], a significant performance reduction over the near-linear scaling for the time-independent case [8]. These issues can be resolved by generalizing the Lie–Trotter–Suzuki product formulae to apply in the time-dependent case. Such formulae have already been developed [11, 12], but have not yet been applied to quantum simulation algorithms. Here we explicitly show how these formulae can be used in quantum algorithms to simulate time-dependent Hamiltonians with complexity near-linear in the simulation time. We provide a number of improvements to further improve the efficiency, and a carefully accounting of the computational resources used in the simulation.

Lloyd was the first to propose an explicit quantum algorithm for simulating Hamiltonian evolution [2]. This algorithm is for systems that are composed of subsystems of limited dimension, with a time-independent Hamiltonian consisting of a sum of interaction terms. The algorithm uses the Trotter formula to express the time evolution operator as a sequence of exponentials of these interaction Hamiltonians, which may be simulated efficiently. As a result, the complexity of the algorithm scales as  $O(\|H\|\Delta t)^2$ , where  $\Delta t$  is the evolution time, and  $\|H\|$  is spectral norm of the Hamiltonian.

Aharonov and Ta-Shma [6] and Childs [7] extended these ideas to apply to Hamiltonians that are sparse, but have no evident tensor product structure. They use graph coloring techniques to decompose the Hamiltonian into a sum of one-sparse Hamiltonians, and use the Trotter formula and the Strang splitting [13], respectively, to write the evolution operator as a sequence of one-sparse exponentials. The resulting sequence of exponentials can then be performed by a quantum computer. The use of higher-order splitting formula reduces the complexity of Child's algorithm to  $O(\|H\|\Delta t)^{3/2}$ , and it was conjectured that even higher-order formulae may lead to near-linear scaling [7].

This hypothesis was verified by Berry, Ahokas, Cleve and Sanders (BACS) [8]. They used Lie–Trotter–Suzuki formulae [14] to generate arbitrarily high-order product formula approximations to the time-evolution operator, and gave an improved method for decomposing the Hamiltonian. The use of the Lie–Trotter–Suzuki formulae reduced the cost of their algorithm, causing it to scale as  $(\|H\|\Delta t)^{1+o(1)}$ . An alternative approach using quantum walks can yield scaling strictly linear in  $\|H\|\Delta t$  [20, 21].

High-order Trotter-like approximations for ordered operator exponentials are needed to extend the results of BACS to apply to the simulation of time-dependent Hamiltonian evolution. Such integrators were originally proposed by Suzuki [11], using a time-displacement superoperator. This method is made rigorous in Ref. [12], where sufficiency criteria for the applicability of the formulae, as well as bounds for the error, are provided.

Here we explicitly apply these integrators to provide an algorithm for simulation of sparse time-dependent Hamiltonians, and find that its complexity depends on the norms of  $H(t)$  and its derivatives. We show how adaptive time steps may be employed such that the complexity depends on average values of these norms, rather than the maximum values. This approach provides improved efficiency in situations where the norms have a sharp peak, or a finite number of discontinuities. For situations with singularities, we show how efficiency may be improved by adapting the order of the integrators.

We also improve the performance by specifying that the oracles that encode the Hamiltonian encode their outputs in polar form. Given this encoding, the one-sparse exponentials may be implemented via a simple circuit. In addition, we improve simulation efficiency by expressing the Hamiltonian as a sum in different bases. We quantify the performance of our scheme by considering the errors that occur in every step of the algorithm, including errors that occur because of discretization of the times used by our quantum oracles. We provide a unified presentation taking account of all these factors, as they interact in nontrivial ways.

## II. OUR APPROACH

In this section, we provide a less technical explanation of the results and how they are obtained. Then we give the rigorous proofs in the following sections. The objective in a quantum simulation algorithm is to simulate evolution under the Schrödinger equation

$$\frac{\partial}{\partial t} |\psi(t)\rangle = -iH(t) |\psi(t)\rangle, \quad (1)$$

where  $H(t)$  is the time-dependent Hamiltonian. That is, the initial state  $|\psi(t_0)\rangle$  is encoded in the qubits of the quantum computer, and we wish to obtain a state in the quantum computer encoding an approximation of the final state  $|\psi(t_0 + \Delta t)\rangle$ . A quantum computer simulation algorithm achieves this by applying an (encoded) approximation of the time-ordered exponential

$$U(t_0 + \Delta t, t_0) = \mathcal{T} \exp \left\{ -i \int_{t_0}^{t_0 + \Delta t} H(u) du \right\}, \quad (2)$$

to the initial state in the quantum computer such that  $|\psi(t_0 + \Delta t)\rangle = U(t_0 + \Delta t, t_0) |\psi(t_0)\rangle$ . Given any  $\epsilon > 0$ , our goal is to obtain an approximation of the final state that is within trace distance  $\epsilon$  of the true state. This can be achieved [8] if the approximation,  $\tilde{U}(t_0 + \Delta t, t_0)$ , satisfies

$$\left\| U(t_0 + \Delta t, t_0) - \tilde{U}(t_0 + \Delta t, t_0) \right\| \leq \epsilon, \quad (3)$$

with  $\|\cdot\|$  defined to be the two-norm.

### A. Constant-Sized Time Step Simulation

Our primary objective in this paper is to demonstrate a quantum simulation algorithm for time-dependent Hamiltonian evolution that has a time-complexity that scales as  $\Delta t^{1+o(1)}$ . The simplest approach to do so involves combining the sparse Hamiltonian decomposition scheme of BACS [8], together with the higher-order integrators from Refs. [11, 12]. BACS show that a Hamiltonian with sparseness parameter (the maximum number of nonzero elements in any nonzero row or column) of  $d$  may be decomposed into  $6d^2$  one-sparse Hamiltonians. Given that the state is encoded on  $n$  qubits, there is an additional factor of  $\log^* n$  to the number of queries required for the simulation. Here  $\log^*$  represents the iterated logarithm function, and increases extremely slowly with  $n$ . This factor arises because the decomposition requires  $O(\log^* n)$  queries to the oracle for the Hamiltonian to perform the decomposition.

The BACS decomposition technique may be used in the time-dependent case. However, one complication is that the sparseness can, in the completely general case, depend on time. That is, the nonzero elements at one time can be different to those at a different time. This would mean that the decomposition depends on the time, which makes the use of Lie–Trotter–Suzuki formulæ problematic. To avoid that problem, we consider every matrix element that *ever* attains a nonzero value to be non-zero and use the BACS decomposition algorithm on those matrix elements. This allows us to directly apply their decomposition result.

Reference [12] gives a result for exponentials of a general operator  $A(t)$ . The result for Hamiltonian evolution follows by taking  $A(t) = iH(t)$ . Then, for  $H(t) = \sum_{j=1}^m H_j(t)$ , by using a Lie–Trotter–Suzuki product formula that is accurate to order  $2k$  [11], simulation error within  $\epsilon$  may be achieved using a number of one-sparse exponentials that scales as

$$O \left( mk \left( \frac{25}{3} \right)^k (\Lambda \Delta t)^{1+1/2k} / \epsilon^{1/2k} \right). \quad (4)$$

Here  $\Lambda$  is an upper bound on the derivatives of the Hamiltonians such that

$$\Lambda \geq \left( \sum_{j=1}^m \|H_j^{(p)}(t)\| \right)^{1/(p+1)}, \quad (5)$$

for  $t \in [t_0, t_0 + \Delta t]$  and  $p \in \{0, \dots, 2k\}$  [25]. The notation with superscript  $(p)$  denotes repeated derivatives. An upper bound is used, rather than the exact maximum value, because the oracles that are used only give matrix elements of the Hamiltonian, not the norm. This is significant because methods for computing the norm of a matrix are often inefficient. However, it is often possible to place an upper bound on the norm, even if it is not possible to determine  $\Lambda$  exactly.

We convert this result into a number of oracle queries by multiplying the number of exponentials by the number of oracle queries that are needed to simulate a one-sparse operator exponential. By doing so, we find that if the Hamiltonian is sufficiently smooth then the query complexity of the algorithm scales as  $(\Lambda \Delta t)^{1+o(1)}$ .

More generally, we also consider the case where  $H(t)$  has discontinuous derivatives at a finite number of times. Such discontinuities are problematic because if a Lie–Trotter–Suzuki formula is used to integrate across such a discontinuity, then error bounds proved in [12] may not apply. In some cases this can be rectified by reducing the order of the integrator, but this strategy is not applicable if the Hamiltonian is not at least twice differentiable. Instead, we choose the time intervals to omit these points of discontinuity. In order to use this approach, it is necessary that the norm of the Hamiltonian is adequately bounded, because otherwise there could be significant evolution of the system very close to the point of discontinuity. Given this restriction it is possible to perform the simulation with complexity that is essentially unchanged. The full result, with the required conditions, is given in Corollary 6.

It is important to note that the performance of our constant step size algorithm scales with the largest possible value of the norms of  $H_j$  and their derivatives. For some Hamiltonians, these values may only be large for a small fraction of the simulated evolution and so the algorithm may be inefficient. Using adaptive time steps, we can overcome this problem and demonstrate complexities that scale with the average values of the norms of  $H_j$  and their derivatives, given additional restrictions on the Hamiltonian are met. We discuss this approach below.

## B. Adaptive time steps

The above scaling is the direct application of the results of Refs. [8] and [12]. We improve this scaling for problems where  $H(t)$  is badly behaved. For example, the matrix  $H(t)$  may be rapidly changing at some times and may be slowly varying at others. In such cases, using constant step size methods may be inefficient because overly conservative time steps will be taken during time intervals in which the Hamiltonian is comparatively easy to simulate. We address this by introducing adaptive time steps. When using adaptive time steps, instead of choosing each time step to be  $\Delta t/r$ , a sequence of times  $\{t_p\}$  are chosen such that  $t_0 < t_1 < \dots < t_r = t_0 + \Delta t$ . The size of the time intervals can be varied, as can the order of the product formula within each interval.

We choose the duration of the time steps using a time-dependent function,  $\Upsilon(t)$ , that provides similar information to  $\Lambda$ , but at a specific time. We express this function as

$$\Upsilon(t) \geq \left( \sum_{j=1}^m \|H_j^{(p)}(t)\| \right)^{1/(p+1)}, \quad (6)$$

for  $p \in \{0, \dots, 2k\}$ . Throughout this work we use the notation for the average value,  $\bar{\Upsilon}(t_b, t_a)$ , defined by

$$\bar{\Upsilon}(t_b, t_a) := \frac{1}{t_b - t_a} \int_{t_a}^{t_b} \Upsilon(t) dt. \quad (7)$$

The goal is to choose the time steps adaptively such that the number of queries depends on the average value of  $\Upsilon(t)$  over the interval, rather than its maximum value (previously denoted  $\Lambda$ ). The full result is given in Theorem 8. The basis of the method is to choose  $r$  time intervals to limit the error within each time interval to be no greater than  $\epsilon/r$ .

To choose these time intervals appropriately, we need to know what the maximum value of  $\Upsilon(t)$  is over a given interval, because the maximum value of  $\Upsilon(t)$  dictates the simulation error over a short time step. Ideally one would want a method of choosing the duration of these steps that depends only on the value of  $\Upsilon(t)$  at the beginning of the interval, in order to avoid needing to know the value of  $\Upsilon(t)$  over the entire interval. We achieve this by requiring that the derivative of  $\Upsilon(t)$  is appropriately bounded. A bound on the derivative is also necessary to obtain a result

depending on the average of  $\Upsilon(t)$ . This is because we can demonstrate that if the value of  $\Upsilon(t)$  is approximately constant during each time step, then the complexity of the algorithm scales with the average value of  $\Upsilon(t)$  over all time steps. We then require that the derivative of  $\Upsilon(t)$  is bounded in order to guarantee this approximate constancy in the limit of short time steps.

Even given the restriction on the derivative, it is unclear how to effectively choose the time intervals. The problem is that the time intervals are chosen to ensure that the error in each interval is no greater than  $\epsilon/r$ , but then the number of intervals will depend on how the intervals were chosen. To break this circular logic, we need a way of choosing an  $r_g$ , such that when we choose the time intervals to have error no greater than  $\epsilon/r_g$ , the total number of intervals is no larger than  $r_g$ . The full technique is given in the proof of Theorem 8. The value of  $r_g$  is chosen as in Eq. (50). The duration of each time step can then be calculated using just information about  $\Upsilon(t)$  at the beginning of the time step, via an increment inversely proportional to  $\Upsilon(t_p)$ , as given in Eq. (51).

### C. Resource Analysis

Our cost analysis of these simulation algorithms focuses on the number of queries that are made to a pair of quantum oracles that provide information about the locations and values of the nonzero matrix elements of the Hamiltonian. We provide improvements to these oracles that enable us to improve the efficiency of the simulation, as well as to more precisely quantify the resource usage. First, we require that the output of the oracle for the values of the matrix is encoded using a qubit string as a complex number in polar form. The advantage of this is that the one-sparse Hamiltonian evolution can be applied using a simple circuit with qubit rotations proportional to the magnitude and phase of the matrix element. This is shown in Sec. VIII, and the explicit circuit is given in Fig. 3. Furthermore, the qubit rotations may be performed independently for each qubit of precision yielded by the oracle.

To more precisely quantify the resource usage, we examine the number of qubits that the oracles need to provide, as well as the number of bits needed to represent the time. The oracles that are traditionally used in quantum simulation algorithms yield many-qubit approximations to the matrix elements in a single query. The fact that the qubits yielded by the oracle may be used independently motivates using oracles that output only one qubit per query. Doing so further reduces the number of qubits needed to simulate a one-sparse Hamiltonian evolution, because qubits that are not currently being used need not be stored.

We find that the total number of qubits accessed for the positions of the nonzero matrix elements scales as  $n \log^* n$  (see Lemma 9), due to the need to access each of the  $n$  qubits for the position. This yields a factor of  $n$  increase in the apparent complexity over that if all qubits can be obtained in a single query. The number of qubits accessed for the values of the matrix elements is independent of  $n$ , but it does depend on other simulation parameters such as the error tolerance, the evolution time, the sparseness of the Hamiltonian, the norm of the derivative of  $H$  and  $k$ . It depends on all of these quantities logarithmically, except for  $k$ . The precise result for the number of oracle queries used is given in Lemma 1.

We also consider a new source of error that is unique to the simulation of time-dependent Hamiltonians: the discretization of the time. Because the Hamiltonian varies with time, inaccuracy in the time will result in inaccuracy in the estimate of the Hamiltonian. This is potentially problematic for Lie-Trotter-Suzuki product formulæ, which rely upon precise times in order to obtain higher-order scaling for the error. The higher-order scaling is not strictly obtained when the time is discretized, so it is necessary to show that the product formulæ are not overly sensitive to error in the time so that we can use our discrete oracle in place of the continuous Hamiltonian. Another source of difficulty is in simulating systems with discontinuities. The problem is that the technique to avoid discontinuities requires choosing times arbitrarily close to, but on one side of, the discontinuity. With discretization of the time, the rounding may yield times on the wrong side of the discontinuity.

In contrast to the output from the oracle, there is no need to use a coherent superposition of times, and the time may be regarded as a purely classical quantity at all stages in the calculation. This means that it is less challenging to provide the time to high accuracy than it is to obtain output from the oracle to high accuracy. Nonetheless, it is important for the reliability of the simulation that it is not unstable with inaccuracy in the time. We find that the simulation is stable with the time precision. The precision required depends on  $k$ ,  $d$ ,  $\Delta t$  and the maximum norm of the derivative of the Hamiltonian. It is logarithmic in all these quantities except for  $k$ ; see Lemma 1 for the full result. The time precision also affects the result for simulating evolution with discontinuities in Corollary 6. That result holds provided the time discretization is no greater than the time between discontinuities (condition 4).

### III. BACKGROUND AND DEFINITIONS

Next we describe the technical background and definitions needed to understand our full results, which are given in the next section. For generality, we consider a Hamiltonian that is not sparse in any known basis, but is the sum of Hamiltonians that are each sparse in their own canonical basis. That is,

$$H(t) = \sum_{\mu=1}^M H'_\mu(t), \quad (8)$$

where the set of operators  $\{H'_\mu : \mathbb{R} \mapsto \mathbb{C}^{N \times N}; \mu = 1, \dots, M\}$  is sparse when represented in their canonical bases. We express  $H'_\mu(t) = T_\mu^\dagger H_\mu(t) T_\mu$ , where  $H_\mu(t)$  is sparse in the computational basis, and  $T_\mu$  is a basis transformation that maps the computational basis to the canonical basis of  $H_\mu(t)$ .

To avoid complications due to the nonzero elements changing as a function of time, we consider only those elements which are nonzero at any time. We then take  $d$  to be an upper bound on the number of elements in any row that are nonzero at any time in the interval of interest:

**Definition 1.** *The set of operators  $\{H_\mu\}$ , where  $H_\mu : \mathbb{R} \mapsto \mathbb{C}^{N \times N}$ , is  $d$ -sparse on  $\mathcal{S} \subseteq \mathbb{R}$  if for each  $\mu$  there are at most  $d$  matrix elements in each row of  $H_\mu(t)$  that attain a nonzero value for any  $t \in \mathcal{S}$ .*

Given a sparse Hamiltonian, BACS [8] provide a method to decompose the Hamiltonian into a sum of  $6d^2$  one-sparse Hamiltonians, and express the evolution as a product of evolutions under each of these one-sparse Hamiltonians. The definition of sparseness we use here is compatible with that of BACS, and therefore each Hamiltonian  $H_\mu(t)$  may be expressed as a sum of one-sparse Hamiltonians. That is,

$$H(t) = \sum_{\mu=1}^M \sum_{j=1}^{6d^2} T_\mu^\dagger H_{\mu,j}(t) T_\mu, \quad (9)$$

where each  $H_{\mu,j}(t)$  is one-sparse.

This decomposition is enabled by a quantum oracle that answers queries about the locations and values of the nonzero matrix elements of the Hamiltonian [6–9]. For the BACS decomposition, the matrix elements of each  $H_{\mu,j}(t)$  in (9) can be calculated using  $O(\log^* n)$  queries to a quantum oracle for  $H_\mu(t)$ . This oracle yields the locations and values of the nonzero matrix elements in a specified row of  $H_\mu$  to arbitrary precision [8]. In this case the cost of computing the matrix elements to sufficient precision is concealed within the definition of the oracle. To explicitly take account of the number of qubits that the oracle must provide, we separate it into two oracles that yield the positions and values of the nonzero matrix elements, and yield one qubit per query. See Sec. V A for the explicit form of the oracles.

To express the evolution under the Hamiltonian as a product of evolutions under the one-sparse Hamiltonians, BACS use the product formulæ of Suzuki [14]. Suzuki first proposed arbitrary-order product formulæ for both time-independent and time-dependent cases [11, 14]. These product formulæ are for operator exponentials, and are not limited to Hamiltonians. Subsequent work by the present authors showed that Suzuki’s approximation method may be less accurate than expected if the operator does not vary sufficiently smoothly with time [12]. That work provided upper bounds for the approximation error, given that the operator does satisfy a smoothness requirement.

In this work, we use the result from Ref. [12] upper bounding the approximation error, with the operators obtained via the decomposition method of BACS. We therefore adopt the terminology “ $P$ -smooth” and “ $\Lambda$ - $P$ -smooth” from Ref. [12]. These are formally stated below.

**Definition 2.** *The set of operators  $\{H_j : j = 1, \dots, m\}$  is  $P$ -smooth on  $\mathcal{I} \subseteq \mathbb{R}$  if, for each  $H_j$ , the quantity  $\max_{p=0, \dots, P} \|H_j^{(p)}(t)\|$  is finite on  $\mathcal{I}$ .*

**Definition 3.** *The set of operators  $\{H_j : j = 1, \dots, m\}$  is  $\Lambda$ - $P$ -smooth on  $\mathcal{I} \subseteq \mathbb{R}$  if  $\{H_j\}$  is  $P$ -smooth and  $\Lambda \geq \left(\sum_{j=1}^m \|H_j^{(p)}(t)\|\right)^{1/(p+1)}$ , for all  $t \in \mathcal{I}$  and  $p \in \{0, 1, \dots, P\}$ .*

The  $P$ -smooth requirement is needed in order to achieve an approximation of a given order, and the  $\Lambda$ - $P$ -smooth requirement is needed to bound the error. In this work we also consider adaptive time steps, and then it is the upper bound on the derivatives as a function of time that is important. We therefore introduce the following definition.

**Definition 4.** *The set of operators  $\{H_j : j = 1, \dots, m\}$  is  $\Upsilon$ - $P$ -pointwise-smooth on the interval  $\mathcal{I} \subseteq \mathbb{R}$ , for  $\Upsilon : \mathbb{R} \mapsto \mathbb{R}$ , if  $\{H_j\}$  is  $P$ -smooth and  $\Upsilon(t) \geq \left(\sum_{j=1}^m \|H_j^{(p)}(t)\|\right)^{1/(p+1)}$ , for all  $t \in \mathcal{I}$  and  $p \in \{0, 1, \dots, P\}$ .*

In addition we adopt the terminology that a set of Hamiltonians is  $\Lambda$ - $\infty$ -smooth if the set is  $\Lambda$ - $2k$ -smooth for every  $k > 0$ . Similarly, we say that a set of Hamiltonians is  $\Upsilon$ - $\infty$ -pointwise-smooth if it is  $\Upsilon$ - $2k$ -pointwise-smooth for every  $k > 0$ .

Provided  $\{H_j\}$  is  $\Lambda$ - $2k$ -smooth, for the Hamiltonian  $H(t) = \sum_{j=1}^m H_j(t)$  the evolution  $U(t_0 + \Delta t, t_0)$  may be approximated via the integrator  $U_k$ , which is given iteratively via [12]

$$\begin{aligned} U_1(t_0 + \Delta t, t_0) &:= \prod_{j=1}^m \exp\{-iH_j(t_0 + \Delta t/2)\Delta t/2\} \prod_{j=m}^1 \exp\{-iH_j(t_0 + \Delta t/2)\Delta t/2\}, \\ U_\ell(t_0 + \Delta t, t_0) &:= U_{\ell-1}(t_0 + \Delta t, t_0 + (1 - s_\ell)\Delta t)U_{\ell-1}(t_0 + (1 - s_\ell)\Delta t, t_0 + (1 - 2s_\ell)\Delta t) \\ &\quad \times U_{\ell-1}(t_0 + (1 - 2s_\ell)\Delta t, t_0 + 2s_\ell\Delta t)U_{\ell-1}(t_0 + 2s_\ell\Delta t, t_0 + s_\ell\Delta t)U_{\ell-1}(t_0 + s_\ell\Delta t, t_0), \end{aligned} \quad (10)$$

with  $s_\ell = 1/(4 - 4^{1/(2\ell-1)})$ . This formula is implied by Suzuki's work [11], but is stated explicitly in [12]. The approximation error is  $O((\Lambda\Delta t)^{2k+1})$ , so this formula is appropriate for short time intervals. For longer  $\Delta t$ , the evolution time may be divided into  $r$  subintervals, resulting in the approximation

$$U(t_0 + \Delta t, t_0) \approx \prod_{\ell=1}^r U_k\left(t_0 + \ell\frac{\Delta t}{r}, t_0 + (\ell-1)\frac{\Delta t}{r}\right). \quad (11)$$

The value of  $r$  is then chosen large enough such that the overall error is no greater than some allowable error,  $\epsilon$ . This choice of  $r$  scales as  $O((\Lambda\Delta t)^{1+1/2k}/\epsilon^{1/2k})$ . More precisely, the error in the product formula will be no greater than  $\epsilon$  if we take

$$r = \left\lceil 2\epsilon^{-1/2k} (2k(5/3)^{k-1}\Lambda\Delta t)^{1+1/2k} \right\rceil, \quad (12)$$

provided that

$$\epsilon \leq (9/10)(5/3)^k \Lambda\Delta t. \quad (13)$$

This result is equivalent to Lemma 5 of [12], after eliminating  $Q_k$  by using the inequalities in Eq. (A.3) of that paper. The overall complexity of the simulation is then proportional to the value of  $r$ .

#### IV. RESULTS

This section formally presents our main results. The major result is an upper bound for the query complexity used to simulate time-dependent Hamiltonian evolution, using adaptive time steps and oracles that cost one query per yielded qubit. In order to quantify the complexity, the primary goal is to bound the error. In simulation schemes for time-dependent Hamiltonians there are three sources of error:

1. integrator error from using  $U_k$ ,
2. error due to using a finite-bit representation of the time, and
3. error due to using a discretized representation of  $H_\mu$ .

To guarantee that the total error in the simulation is  $\epsilon$ , we ensure that the latter two errors sum to at most half the total error tolerance. The time steps are then chosen such that the contribution to the error from the integrators at most adds up to the remaining half. The following lemma, proved in Appendix A, yields upper bounds for the number of bits of precision for these quantities that are needed to ensure that the roundoff errors are at most  $\epsilon/2$ .

**Lemma 1.** *Let  $\{H_\mu : \mathbb{R} \mapsto \mathbb{C}^{2^n \times 2^n}; \mu = 1, \dots, M\}$  be a set of time-dependent Hermitian operators that is 2-smooth and  $d$ -sparse on  $\mathcal{I}$ , which we take to be the union of disjoint closed intervals  $\{\mathcal{I}_j\}$ . Furthermore define  $\Delta t := \sup_{t_i, t_f \in \mathcal{I}} (t_f - t_i)$ . The round-off error in simulating the product of the time-evolution operators that are generated by  $H(t) = \sum_\mu T_\mu^\dagger H_\mu(t) T_\mu$  over each  $\mathcal{I}_j$ , using the integrator  $U_k$  is  $\epsilon/2$  if*

1. the number of bits of precision used to represent the time,  $n_t$ , and the number of qubits of precision used to represent the matrix elements,  $n_H$ , satisfy

$$\begin{aligned} n_t &\geq \left\lceil \log_2 \left( \frac{(\max_{t \in \mathcal{I}, \mu} \|\partial_t H_\mu(t)\|)(32kMd^2)(5/3)^{k-1}\Delta t^2}{\epsilon} \right) \right\rceil, \\ n_H &\geq 2 \left\lceil \log_2 \left( \frac{32kMd^2(5/3)^{k-1}\Lambda_{\max}\Delta t}{\epsilon} \right) \right\rceil + 6, \end{aligned} \quad (14)$$

2. and for each  $j$ , the length of the subinterval  $\mathcal{I}_j$  obeys

$$|\mathcal{I}_j| \geq \Delta t / 2^{n_t}, \quad (15)$$

where  $\Lambda_{\max}$  is an upper bound on  $\max_{t \in \mathcal{I}, \mu} \|H_\mu(t)\|$ .

Given these conditions on the number of bits of precision, we can then bound the number of queries to simulate the Hamiltonian using adaptive time steps as in the following theorem.

**Theorem 2.** *If  $\{H_\mu : \mathbb{R} \mapsto \mathbb{C}^{2^n \times 2^n}; \mu = 1, \dots, M\}$  is a set of time-dependent Hermitian operators that is  $\Upsilon$ - $2k$ -pointwise-smooth,  $d$ -sparse on  $\mathcal{I} = [t_0, t_0 + \Delta t]$ ,  $n_t$ , and  $n_H$  satisfy (14), and there exists  $K \in \mathbb{R}$  such that  $|\partial_t \Upsilon(t)| \leq K^2 [\Upsilon(t)]^2 \forall t \in \mathcal{I}$ , then for any  $\epsilon \in (0, 1]$ , the evolution generated by  $H(t) = \sum_\mu T_\mu^\dagger H_\mu(t) T_\mu$  can be simulated within error  $\epsilon$ , and with the number of queries (denoted  $QueryCost$ ) to our Hamiltonian oracles, satisfying*

$$QueryCost \in O\left([n \log^* n + \log(kM(5/3)^k d^2 \Lambda_{\max} \Delta t / \epsilon)] N_{\text{exp}}\right), \quad (16)$$

$$N_{\text{exp}} \in O\left(M d^2 k (25/3)^k \frac{[d^2 \tilde{\Upsilon}(t_0 + \Delta t, t_0) \Delta t]^{1+1/2k}}{\epsilon^{1/2k}}\right), \quad (17)$$

where  $\log^*$  is the iterated logarithm function and  $\Lambda_{\max}$  is given in Lemma 1. The number of calls to  $\{T_\mu\}$ , namely  $N_T$ , satisfies  $N_T \in O(N_{\text{exp}} / (3d^2))$ .

This result is stated in asymptotic (big-O) notation wherein we take

$$\tilde{\Upsilon}(t_0 + \Delta t, t_0) \Delta t, \Lambda_{\max} \Delta t, \max_{t \in \mathcal{I}, \mu} \|\partial_t H_\mu(t)\| \Delta t^2, M, d, n, \epsilon^{-1}, k \quad (18)$$

to be our asymptotic parameters. Specifically, we say for two functions of these parameters,  $f$  and  $g$ , that  $f \in O(g)$  if there exists a constant  $a > 0$  such that  $|f| \leq a|g|$  if *all* of these parameters are sufficiently large.

If  $H(t)$  is the sum of sufficiently smooth terms and bounded on  $t \in [t_0, \infty)$ , then  $k$  can be chosen such that (16) and (17) scale nearly linearly with  $\tilde{\Upsilon}(t_0 + \Delta t, t_0) \Delta t$ . This observation is important because linear scaling is known to be a lower bound, and therefore our time scaling is nearly optimal [8, 22]. This observation is stated formally in the following corollary.

**Corollary 3.** *If, in addition to the requirements of Theorem 2,  $\{H_\mu\}$  is  $\Upsilon$ - $\infty$ -pointwise-smooth and  $d$ -sparse on  $[t_0, \infty)$  then,*

$$QueryCost \in [n \log^* n + n_H] M d^4 \tilde{\Upsilon}(t_0 + \Delta t, t_0) \Delta t (d^2 \tilde{\Upsilon}(t_0 + \Delta t, t_0) \Delta t / \epsilon)^{o(1)}. \quad (19)$$

Before proceeding to show how to perform simulations with adaptive time steps, we first give the explicit scheme without adaptive time steps, and show how to take account of discontinuities in the Hamiltonian.

## V. SIMULATING TIME-DEPENDENT HAMILTONIANS

In this section, we show how to simulate time-dependent Hamiltonian evolution on a quantum computer. In particular, we show that sparse time-dependent Hamiltonian evolution can be simulated efficiently provided that the Hamiltonian is at least piecewise twice-differentiable. We also show that if  $H(t)$  is sufficiently smooth, then the query complexity of our simulation scheme is comparable to the cost of the BACS algorithm for simulating time-independent Hamiltonian evolution. First we give the explicit description of the oracles used, then we give the precise result for the complexity of the simulation in terms of these oracles.

### A. Oracle Calls

The time-dependent Hamiltonian over an  $N$ -dimensional Hilbert space  $\mathcal{H}$  can be represented by a matrix with elements  $H_{xy}$ , for  $x$  the row number and  $y$  the column number. We consider a quantum oracle that can be queried to provide information about the locations and values of the nonzero matrix elements. For additional generality, in this work we assume that the Hamiltonian is in the form

$$H(t) = \sum_{\mu=1}^M T_\mu^\dagger H_\mu(t) T_\mu, \quad (20)$$

where  $\{H_\mu\}$  is  $d$ -sparse. This takes account of cases where the overall Hamiltonian is not sparse, but it may still be simulated efficiently using efficient basis transformations  $T_\mu$  [4, 5, 10, 20]. We therefore require oracles to give the locations and values of the nonzero elements of the matrix representation of each  $H_\mu(t)$ .

Our first oracle provides the column numbers of the nonzero matrix elements in a given row of any  $H_\mu$ . The function yields a requested bit of a particular entry in a list of  $d$  column numbers. This list contains the column number of every matrix element in a specified row of  $H_\mu$  that attains a nonzero value. This function is **Col**, where **Col**( $p, i, x, \mu$ ) yields the  $p^{\text{th}}$  bit of the  $i^{\text{th}}$  potentially nonzero matrix element in row  $x$  of  $H_\mu$ .

Our second oracle provides a requested matrix element of  $H_\mu(t)$ . This function yields a requested bit of a binary encoding of a given matrix element evaluated at a specified time. We denote this function as **MatrixVal**, and define **MatrixVal**( $p, x, y, \mu, q$ ) to yield the  $p^{\text{th}}$  bit of a binary encoding of the matrix element  $[H_\mu(t_q)]_{xy}$ . To take account of discretization of the time, the time is specified by an integer  $q$ , which gives a time from a finite mesh  $\{t_q\}$ , with

$$t_q = t_0 + (q - 1/2)\Delta t/2^{n_t}, \quad (21)$$

where  $n_t$  is a positive integer and  $[t_0, t_0 + \Delta t]$  contains the simulation time interval. We choose the matrix elements to be encoded in polar form,  $(H_\mu(t))_{xy} = \rho(t)\exp(i\phi(t))$ , where  $\rho$  and  $\phi$  are real numbers. For convenience, we also assume that  $\rho(t)$  is encoded as

$$\Lambda_{\max} \left( \frac{\rho_1}{2} + \frac{\rho_2}{2^2} + \dots + \frac{\rho_{n_H}}{2^{n_H}} \right), \quad (22)$$

where  $\rho_j$  refers to the  $j^{\text{th}}$  bit of  $\rho$ , and  $\Lambda_{\max}$  is an upper bound for  $\max_{t \in \mathcal{I}} \max_{\mu=1 \dots M} \|H_\mu(t)\|_{\max}$ .

The quantum oracles are unitary operations that give the values of these functions. That is, for classical inputs  $p$ ,  $i$  and  $\mu$  and the quantum input  $|x\rangle$ ,

$$\mathbf{QCol}(p, i, \mu) |x\rangle |0\rangle = |x\rangle |\mathbf{Col}(p, i, x, \mu)\rangle. \quad (23)$$

This differs from Ref. [8], where  $i$  was given as a quantum input. In that work no superposition over the  $i$  was needed, so it does not change the analysis to give it as a classical input. Similarly, for classical inputs  $p$ ,  $\mu$  and  $q$  and the quantum inputs  $|x\rangle$  and  $|y\rangle$ ,

$$\mathbf{QMatrixVal}(p, \mu, q) |x\rangle |y\rangle |0\rangle = |x\rangle |y\rangle |\mathbf{MatrixVal}(p, x, y, \mu, q)\rangle. \quad (24)$$

In the following section, we present asymptotic estimates of the query complexity for simulating time-dependent Hamiltonian evolution using a sequence of approximations  $U_k$ , which are implemented on a quantum computer equipped with oracles **QMatrixVal**, **QCol** and  $\{T_\mu\}$ . We will quantify the number of calls to **QMatrixVal** and **QCol** together as  $QueryCost$ , and quantify the number of calls to  $\{T_\mu\}$  separately as  $N_T$ .

## B. Constant Timestep Simulation Method

The simulation problem is as follows. Given the oracles **QMatrixVal**, **QCol** and the set of oracles  $\{T_\mu\}$ , we wish to simulate the evolution generated by the Hamiltonian given in (20), for  $\{H_\mu\}$   $\Lambda$ - $2k$ -smooth and  $d$ -sparse. In addition, the user is provided with an upper bound for the norm of each  $H_\mu(t)$  and a similar upper bound for the norms of its derivatives. Our task is to provide an upper bound for the number of oracle queries that are needed to simulate the evolution generated by  $H(t)$  within error  $\epsilon$ . These upper bounds are given in the following lemma.

**Lemma 4.** *If  $\{H_\mu : \mathbb{R} \mapsto \mathbb{C}^{2^n \times 2^n}; \mu = 1, \dots, M\}$  is a set of time-dependent Hermitian operators that is  $\Lambda$ - $2k$ -smooth and  $d$ -sparse on  $[t_0, t_0 + \Delta t]$ , then for any  $\epsilon \in (0, 1]$  the evolution generated by  $H(t) = \sum_\mu T_\mu^\dagger H_\mu(t) T_\mu$  can be simulated with the error due to the integrator bounded by  $\epsilon/2$ , and a number of queries to **QCol** and **QMatrixVal** satisfying*

$$QueryCost \leq 12CMd^2 5^{k-1} \left[ 24kd^2 \Lambda \Delta t \left( \frac{5}{3} \right)^k \left( \frac{6d^2 \Lambda \Delta t}{\tilde{\epsilon}/2} \right)^{1/2k} \right], \quad (25)$$

where  $\tilde{\epsilon} := \min\{\epsilon, 18(5/3)^{k-1} d^2 \Lambda \Delta t\}$ ,  $C$  is the number of oracle calls needed to simulate a one-sparse Hamiltonian. The number of queries to  $\{T_\mu\}$ ,  $N_T$ , is bounded above by  $QueryCost/(3Cd^2)$ .

*Proof.* Our approach is to express  $H(t)$  as a sum of  $6Md^2$  one-sparse Hamiltonians and use (11) to approximate  $U(t_0 + \Delta t, t_0)$  with a sequence of these one-sparse operator exponentials. The number of oracle calls in the simulation is then obtained by multiplying the number of one-sparse exponentials in our approximation by the number of oracle



calls to simulate one-sparse Hamiltonian evolution. That number is just given as  $C$  here, and an upper bound will be placed on it in Lemma 9.

Each  $d$ -sparse Hamiltonian  $H_\mu$  may be decomposed into a sum of  $6d^2$  one-sparse Hamiltonians  $H_{\mu,j}$  by using the BACS decomposition scheme. As each matrix element of  $H_\mu$  is uniquely assigned to a one-sparse Hamiltonian  $H_{\mu,j}$  in the BACS decomposition [8], we have that  $\|H_{\mu,j}^{(p)}(t)\| \leq \|H_\mu^{(p)}(t)\|$  for any non-negative integer  $p \leq 2k$ . Using Definition 3, if  $\{H_\mu\}$  is  $\Lambda$ - $2k$ -smooth, then the set of Hamiltonians  $\{H_{\mu,j}\}$  is  $6d^2\Lambda$ - $2k$ -smooth. Using Eq. (12) and the fact that  $N_{\text{exp}} = 2m5^{k-1}r$ , if  $\epsilon/2 \leq (9/10)(5/3)^k 6d^2\Lambda\Delta t$ , then the number of exponentials needed to simulate Hamiltonian evolution, using constant-sized time steps and within error  $\epsilon/2$ , is bounded above by

$$N_{\text{exp}} \leq 2m5^{k-1} \left[ \frac{2[2k(5/3)^{k-1}6d^2\Lambda\Delta t]^{1+1/2k}}{(\epsilon/2)^{1/2k}} \right]. \quad (26)$$

Note that we have replaced  $\epsilon$  with  $\epsilon/2$ , because we require error bounded by  $\epsilon/2$  here. To ensure that condition (13) holds, we then replace  $\epsilon$  by  $\tilde{\epsilon}$ , which ensures that this condition holds and that the error is no greater than  $\epsilon/2$ .

Using  $m = 6Md^2$  and simplifying gives

$$N_{\text{exp}} \leq 12Md^25^{k-1} \left[ 24kd^2\Lambda\Delta t \left(\frac{5}{3}\right)^k \left(\frac{6d^2\Lambda\Delta t}{\tilde{\epsilon}/2}\right)^{1/2k} \right]. \quad (27)$$

The number of oracle queries that are needed in our simulation is  $QueryCost = CN_{\text{exp}}$ , which gives Eq. (25).

Finally, we verify the claim that the number of basis transformations is bounded above by  $N_{\text{exp}}/(3d^2)$  by counting the number of basis transformations that result from using the Lie-Trotter-Suzuki formula. As the BACS decomposition method expresses a  $d$ -sparse Hamiltonian as a sum of  $6d^2$  one-sparse Hamiltonians, and  $\{H_\mu\}$  is  $d$ -sparse, it follows that the Hamiltonian can be expressed as

$$H(t) = \sum_{\mu=1}^M T_\mu^\dagger \left( \sum_{j=1}^{6d^2} H_{\mu,j}(t) \right) T_\mu, \quad (28)$$

where  $H_\mu(t) = \sum_{j=1}^{6d^2} H_{\mu,j}(t)$ , and  $\{H_{\mu,j}\}$  is one-sparse. We can then use (10) to show that  $U_1(t_0 + \tau, t_0)$  becomes

$$\left[ \prod_{\mu=1}^M T_\mu^\dagger \left( \prod_{j=1}^{6d^2} \exp(-iH_{\mu,j}(t_0 + \tau/2)\tau/2) \right) T_\mu \right] \left[ \prod_{\mu=M}^1 T_\mu^\dagger \left( \prod_{j=6d^2}^1 \exp(-iH_{\mu,j}(t_0 + \tau/2)\tau/2) \right) T_\mu \right]. \quad (29)$$

Equation (29) has only  $4M$  basis transformations, but  $12Md^2$  one-sparse operator exponentials. Because  $U_k$  is a product of  $5^{k-1}$  such approximations, there are at most  $3d^2$  basis transformations per one-sparse operator exponential in  $U_k$ . Because the approximation to  $U$  in our decomposition is a product of  $U_k$  [12],

$$N_T \leq N_{\text{exp}}/(3d^2), \quad (30)$$

which implies that  $N_T \leq QueryCost/(3Cd^2)$  via this method.  $\square$

This Lemma shows how the complexity scales when the Hamiltonian is permitted to be a sum of terms that are each individually sparse in different bases. This result only considers the error in integrator, and does not consider the contribution of the round-off error that occurs due to discretizing both time and the matrix elements of  $H_\mu$ . The following Theorem gives values of the precision that are sufficient to ensure this round-off error is  $\epsilon/2$ , implying that the total error is at most  $\epsilon$ .

**Theorem 5.** *If  $\{H_\mu : \mathbb{R} \mapsto \mathbb{C}^{2^n \times 2^n}; \mu = 1, \dots, M\}$  is a set of time-dependent Hermitian operators that is  $d$ -sparse and  $\Lambda$ - $2k$ -smooth on  $[t_0, t_0 + \Delta t]$ , and  $n_t$  and  $n_H$  satisfy (14), then for any  $\epsilon \in (0, 1]$ , the evolution generated by  $H(t) = \sum_{\mu} T_\mu^\dagger H_\mu(t) T_\mu$  can be simulated within error  $\epsilon$ , while using a number of oracle queries to **QMatrixVal** and **QCol**,  $QueryCost$ , that are bounded above by*

$$QueryCost \leq 12CMd^25^{k-1} \left[ 24kd^2\Lambda\Delta t \left(\frac{5}{3}\right)^k \left(\frac{6d^2\Lambda\Delta t}{\tilde{\epsilon}/2}\right)^{1/2k} \right], \quad (31)$$

where  $\tilde{\epsilon} := \min\{\epsilon, 18(5/3)^{k-1}d^2\Lambda\Delta t\}$  and the number of queries to  $\{T_\mu\}$ ,  $N_T$ , obeys  $N_T \leq QueryCost/(3Cd^2)$ .

*Proof.* The error in our simulation scheme arises from two sources: the discretization error, and the error due to the integrator. By requiring that  $n_t$  and  $n_H$  satisfy (14), Lemma 1 implies that the error due to the discretization is no more than  $\epsilon/2$ . Note that, because  $\mathcal{I}$  is just a single time interval, the  $\Delta t$  here corresponds to the  $\Delta t$  in Lemma 1, and the condition (15) is automatically satisfied. Using Lemma 4, we find that the simulation can be performed such that the error in the integrator is no greater than  $\tilde{\epsilon}/2$ , and the query complexity of the simulation satisfies the inequalities in Eqs. (25) and (30). Because  $\tilde{\epsilon} \leq \epsilon$ , using the triangle inequality shows that the total error is no greater than  $\epsilon$ .  $\square$

In some cases where  $\{H_\mu\}$  is not smooth, the Hamiltonian evolution can be simulated more efficiently by deleting a neighborhood from  $[t_0, t_0 + \Delta t]$  around each point where the derivatives of  $\{H_\mu\}$  diverge, and using the integrator  $U_k$  to approximate the time-evolution in the remainder of the interval. The query complexity for simulating Hamiltonian evolution by this method is given by the following corollary.

**Corollary 6.** *Let  $\{H_\mu : \mathbb{R} \mapsto \mathbb{C}^{2^n \times 2^n}; \mu = 1, \dots, M\}$  be a set of time-dependent Hermitian operators that is  $\Lambda$ - $2k$ -smooth on  $\mathcal{I} = (t_0, t_0 + \Delta t) \setminus \{t_1, \dots, t_L\}$ , where  $t_0 < t_1 < \dots < t_L < t_0 + \Delta t$ , with the additional conditions*

1.  $\exists H_{\max} \in \mathbb{R} : H_{\max} \geq \max_{t \in [t_0, t_0 + \Delta t]} \|H(t)\|$ ,
2.  $0 < \epsilon \leq \min\{1, 27(5/3)^{k-1} d^2 \Lambda \Delta t\}$ ,
3.  $n_t$  and  $n_H$  satisfy (14), and
4.  $\Delta t / 2^{n_t} < \min_{\ell=0, \dots, L} (t_{\ell+1} - t_\ell)$  with  $t_{L+1} := t_0 + \Delta t$ .

Then the query complexity for simulating evolution generated by  $H(t) = \sum_\mu T_\mu^\dagger H_\mu(t) T_\mu$  within an error of  $\epsilon$  is

$$\text{QueryCost} \leq 12CMd^2 5^{k-1} \left[ (L+1) + 24kd^2 \Lambda \Delta t \left( \frac{5}{3} \right)^k \left( \frac{6d^2 \Lambda \Delta t}{(\epsilon/3)} \right)^{1/2k} \right], \quad (32)$$

where  $C$  is the number of oracle calls needed to simulate a one-sparse Hamiltonian, and  $N_T \leq \text{QueryCost}/(3Cd^2)$ .

*Proof.* We remove  $\delta$ -neighborhoods around each  $t_\ell$  and simulate evolution on the remaining time interval. We then exploit the fact that  $H(t)$  is bounded to estimate the error incurred by omitting the evolution around those points.

We choose a value of  $\delta$  satisfying

$$0 < \delta \leq \min \left\{ \frac{1}{2} \left[ \min_{\ell=0, \dots, L} (t_{\ell+1} - t_\ell) - \Delta t / 2^{n_t} \right], \frac{1}{2H_{\max}} \log \left( 1 + \frac{\epsilon/6}{L+2} \right) \right\}, \quad (33)$$

where  $t_{L+1} := t_0 + \Delta t$ . The evolution operator is

$$U(t_0 + \Delta t, t_0) = U(t_{L+1}, t_{L+1} - \delta) \left[ \left( \prod_{\ell=L}^1 U(t_{\ell+1} - \delta, t_\ell + \delta) U(t_\ell + \delta, t_\ell - \delta) \right) \times U(t_1 - \delta, t_0 + \delta) U(t_0 + \delta, t_0) \right] \quad (34)$$

$$\approx \prod_{\ell=L}^0 U(t_{\ell+1} - \delta, t_\ell + \delta). \quad (35)$$

For  $\epsilon \leq 27(5/3)^{k-1} d^2 \Lambda \Delta t$ ,

$$2\epsilon(t_{\ell+1} - t_\ell - 2\delta)/(3\Delta t) \leq 18(5/3)^{k-1} d^2 \Lambda (t_{\ell+1} - t_\ell - 2\delta). \quad (36)$$

With this restriction, using Lemma 4 each of the  $L+1$  evolutions in (35) can be simulated with integrator error bounded above by  $\epsilon(t_{\ell+1} - t_\ell - 2\delta)/(3\Delta t)$  using no more than

$$12CMd^2 5^{k-1} \left[ 24kd^2 \Lambda (t_{\ell+1} - t_\ell - 2\delta) \left( \frac{5}{3} \right)^k \left( \frac{6d^2 \Lambda \Delta t}{(\epsilon/3)} \right)^{1/2k} \right] \quad (37)$$

oracle queries. Using Eq. (4.69) of Ref. [15], which states that for unitary operators,

$$\left\| \prod_j U_j - \prod_k V_k \right\| \leq \sum_j \|U_j - V_j\|, \quad (38)$$

the total error is bounded above by

$$\sum_{\ell} \epsilon(t_{\ell+1} - t_{\ell} - 2\delta)/(3\Delta t) < \epsilon/3. \quad (39)$$

Then, summing (37) over  $\ell$  gives inequality (32) as an upper bound for the number of oracle queries made. The additional factor of  $(L + 1)$  in Eq. (32) is to take account of the ceiling function in Eq. (37). The bound for the number of basis transformations is obtained by using  $N_T = N_{\text{exp}}/3d^2$ , and by summing over all  $L + 1$  subintervals.

To bound the overall error, we need to bound the error due to approximating (34) with (35). Because  $H(t)$  is bounded on  $[t_0, t_0 + \Delta t]$ , and  $H_{\text{max}} \geq \max_{t \in [t_0, t_0 + \Delta t]} \|H(t)\|$ , the unitary evolution over each  $t_{\ell}$  for  $\ell = 1, \dots, L$  satisfies

$$\|U(t_{\ell} + \delta, t_{\ell} - \delta) - \mathbb{1}\| \leq e^{2H_{\text{max}}\delta} - 1. \quad (40)$$

For  $\ell = 0$  and  $L + 1$  we have

$$\begin{aligned} \|U(t_0 + \delta, t_0) - \mathbb{1}\| &\leq e^{H_{\text{max}}\delta} - 1, \\ \|U(t_{L+1}, t_{L+1} - \delta) - \mathbb{1}\| &\leq e^{H_{\text{max}}\delta} - 1. \end{aligned} \quad (41)$$

These errors can be made suitably small by using the restriction on  $\delta$  specified in Eq. (33). The first expression in the braces in (33) ensures that the inequality  $\delta < \min_{\ell=0, \dots, L} (t_{\ell+1} - t_{\ell})/2$  is satisfied. The second ensures that the error in approximating the evolution about each of the  $t_{\ell}$  by  $\mathbb{1}$  is bounded above by  $\epsilon/[6(L + 2)]$ . Using Eq. (38), the total error in approximating (34) with (35) is bounded above by  $\epsilon/6$ . Combining this with the bound on the error of  $\epsilon/3$  for the integrators over the time intervals  $[t_{\ell} + \delta, t_{\ell+1} - \delta]$ , the total error due to omitting the  $\delta$ -neighborhoods from the simulation and using the Lie-Trotter-Suzuki formula is bounded above by  $\epsilon/2$ .

We now use Lemma 1 to ensure that the roundoff error is also bounded above by  $\epsilon/2$ . The definition of  $\Delta t$  used in that Lemma gives the  $\Delta t$  used in this corollary, and so may be used in the restrictions without change. The restriction  $\Delta t/2^{n_t} < \min_{\ell} (t_{\ell+1} - t_{\ell})$  and the choice of  $\delta$  in (33) ensure that the restriction (15) of Lemma 1 holds. We have also required that  $n_t$  and  $n_H$  satisfy (14), so all conditions required for Lemma 1 hold, and the round-off error may be bounded by  $\epsilon/2$ . As the error in the integrator has also been bounded by  $\epsilon/2$ , the total error is no greater than  $\epsilon$ .  $\square$

We have shown in this section that time-dependent Hamiltonian evolutions can be simulated by using the product formula approach to simulation, even if there are discontinuities in the Hamiltonian. If the Hamiltonian is sufficiently smooth, then these simulations achieve the same near-linear scaling as the BACS simulation achieves. In the next section we improve upon these results by presenting a method that uses adaptive time steps.

## VI. ADAPTIVE DECOMPOSITION SCHEME

We saw in the previous section that if we use time steps that have constant size in our simulation algorithm, then the number of oracle calls used to simulate Hamiltonian evolution depends on the largest values of the norms of  $H(t)$  and its derivatives. For Hamiltonians whose time-dependence is sharply peaked, the maximum values of these quantities can be quite large relative to their time-averages. It is natural to ask if the complexity can be made to depend on the average values, rather than the maximum values, by using adaptive time steps. In this section we show that this is indeed possible.

As discussed in Sec. II B, this result is nontrivial because of the interdependence of the different parameters. We choose a sequence of times  $\{t_p\}$  such that  $t_0 < t_1 < \dots < t_r = t_0 + \Delta t$ . These times are selected such that the error from using  $U_k$  is bounded above by  $\epsilon/(2r)$  for each interval. Given a function  $\Upsilon$ , such that  $\{H_{\mu}\}$  is  $\Upsilon$ - $2k$ -pointwise-smooth on  $[t_0, t_0 + \Delta t]$ , the size of the interval will be inversely proportional to the maximum value of  $\Upsilon(t)$  in that interval. The exact result is given in the following Lemma.

**Lemma 7.** *Let  $\{H_{\mu} : \mu = 1, \dots, M\}$ , where  $H_{\mu} : \mathbb{R} \mapsto \mathbb{C}^{2^n \times 2^n}$ , be a set of time-dependent Hermitian operators that is  $\Upsilon$ - $2k$ -pointwise-smooth and  $d$ -sparse on  $[t_0, t_0 + \Delta t]$ , and  $\{t_1, \dots, t_r\}$  be a set of  $r$  moments in time such that  $t_0 < t_1 < \dots < t_r = t_0 + \Delta t$ . If*

$$\left( \max_{t \in [t_p, t_{p+1}]} \Upsilon(t) \right) (t_{p+1} - t_p) \leq \frac{(\epsilon/r)^{1/(2k+1)}}{24d^2k(5/3)^{k-1}}, \quad (42)$$

where  $\epsilon \in (0, 1]$ , then

$$\left\| U(t_0 + \Delta t, t_0) - \prod_{p=1}^r U_k(t_p, t_{p-1}) \right\| \leq \epsilon/2. \quad (43)$$

*Proof.* Using Theorem 3 and Eq. (A.3) of Ref. 12, the error in each interval is bound above by

$$\|U(t_p + \Delta t_p, t_p) - U_k(t_p + \Delta t_p, t_p)\| \leq 2[2k(5/3)^{k-1}\Lambda\Delta t_p]^{2k+1}, \quad (44)$$

where  $\Delta t_p := t_{p+1} - t_p$ , provided that

$$4k\sqrt{2}(5/3)^{k-1}\Lambda\Delta t_p \leq 3/2, \quad (45)$$

for a set of operators that is  $\Lambda$ - $2k$ -smooth. As  $\Lambda$  is bounded above by  $6d^2 \max_{t \in [t_p, t_{p+1}]} \Upsilon(t)$ , the upper bound in (44) becomes

$$\|U(t_p + \Delta t_p, t_p) - U_k(t_p + \Delta t_p, t_p)\| \leq 2 \left[ 12d^2 k(5/3)^{k-1} \max_{t \in [t_p, t_{p+1}]} \Upsilon(t) \Delta t_p \right]^{2k+1}. \quad (46)$$

Because  $\epsilon \leq 1$  and  $r \geq 1$ , the condition (42) implies that (45) is satisfied, and therefore that Eq. (46) holds.

Using Eq. (42) in Eq. (46) gives

$$\|U(t_p + \Delta t_p, t_p) - U_k(t_p + \Delta t_p, t_p)\| \leq \epsilon/(2r). \quad (47)$$

Then, using Eq. (38) the error due to using  $r$  Lie-Trotter-Suzuki integrators is at most  $\epsilon/2$ , hence yielding (43).  $\square$

There are two challenges in using this result as a method for choosing the time steps. First, Eq. (42) depends on the maximum value of  $\Upsilon(t)$  over the interval, which may be difficult to find, particularly because it depends on the size of the time interval. Ideally, it should depend only on the value at time  $t_p$ , in order to give a simple method of determining the time interval. Second, Eq. (42) depends on the number of time steps  $r$ , which is not known in advance. It is only known once all time intervals have been determined, but we wish to determine these via Eq. (42).

To break these interdependencies, we first need a bound on the derivative of  $\Upsilon(t)$ , so we can bound the value of  $\Upsilon(t)$  on the interval by the value at time  $t_p$ . It is not obvious what bound should be taken, but it can be shown that if  $\Upsilon_P$  is the smallest possible function such that  $\{H_\mu\}$  is  $\Upsilon$ - $P$ -pointwise-smooth, then it will satisfy  $|\partial_t \Upsilon_P(t)| \leq [\Upsilon_{P+1}(t)]^2$  (see Appendix B). In many cases we can expect that there exists a constant  $K$  such that  $\Upsilon_{P+1}(t) \leq K\Upsilon_P(t)$ . This motivates taking the condition  $|\partial_t \Upsilon(t)| \leq K^2[\Upsilon(t)]^2$ .

Second, we provide a quantity  $r_g$ , given in Eq. (50), that is an upper bound on the number of time steps. If the time steps are chosen according to Eq. (42) with  $r_g$  instead of  $r$ , then Eq. (42) must still hold. Using this approach, we obtain the following Theorem.

**Theorem 8.** *If  $\{H_\mu : \mathbb{R} \mapsto \mathbb{C}^{2^n \times 2^n}; \mu = 1, \dots, M\}$  is a set of time-dependent Hermitian operators that is  $\Upsilon$ - $2k$ -pointwise-smooth,  $d$ -sparse on  $\mathcal{I} = [t_0, t_0 + \Delta t]$ ,  $n_t$ , and  $n_H$  satisfy (14), and there exists  $K \in \mathbb{R}$  such that  $|\partial_t \Upsilon(t)| \leq K^2[\Upsilon(t)]^2 \forall t \in \mathcal{I}$ , then for any  $\epsilon \in (0, 1]$ , the evolution generated by  $H(t) = \sum_\mu T_\mu^\dagger H_\mu(t) T_\mu$  can be simulated within error  $\epsilon$ , with*

$$\text{QueryCost} \leq 12CMd^2 5^{k-1} \left[ \frac{[24d^2 k(5/3)^{k-1} \overline{\Upsilon}(t_0 + \Delta t, t_0) \Delta t]^{1+1/2k}}{(\epsilon/4)^{1/2k}} + 3K^2 \overline{\Upsilon}(t_0 + \Delta t, t_0) \Delta t + 1 \right], \quad (48)$$

where  $\log^*$  is the iterated logarithm function,  $\Lambda_{\max}$  is given in Lemma 1,  $C$  is the number of oracle calls needed to simulate a one-sparse Hamiltonian, and  $N_T \leq \text{QueryCost}/(3Cd^2)$ .

*Proof.* For this proof, we choose a set of  $r$  times  $\{t_1, \dots, t_r\}$ , such that  $t_0 < t_1 < \dots < t_r = t_0 + \Delta t$ . We define

$$A := \frac{[24d^2 k(5/3)^{k-1} \overline{\Upsilon}(t_0 + \Delta t, t_0) \Delta t]^{1+1/2k}}{\epsilon^{1/2k}}, \quad (49)$$

and

$$r_g := \lceil A + 3K^2 \overline{\Upsilon}(t_0 + \Delta t, t_0) \Delta t + 1 \rceil. \quad (50)$$

We choose the times via the recurrence relation, for  $p = 0, \dots, r-1$ ,

$$t_{p+1} = t_p + \frac{1}{\Upsilon(t_p)} \frac{1}{Y + K^2}, \quad (51)$$

where

$$Y := \frac{24d^2k(5/3)^{k-1}}{(\epsilon/r_g)^{1/(2k+1)}}. \quad (52)$$

We will show shortly that (51) implies that (42) is held, thereby guaranteeing that the integrator error is bounded above by  $\epsilon/2$ . In order to prove this we first need to find an upper bound for  $\Upsilon(t + \delta t)$ , for small values of  $\delta t > 0$ , in terms of  $\Upsilon(t)$ .

We find this bound by solving the differential equations given by  $\Upsilon$  saturating the inequality  $|\partial_t \Upsilon(t)| \leq K^2[\Upsilon(t)]^2$ . In particular, for  $\delta t \geq 0$ , we have

$$\frac{\Upsilon(t)}{1 + K^2 \Upsilon(t) \delta t} \leq \Upsilon(t + \delta t) \leq \frac{\Upsilon(t)}{1 - K^2 \Upsilon(t) \delta t}. \quad (53)$$

By using the inequality on the left, and the fact that the minimum of a function is less than or equal to its average value, we find

$$\overline{\Upsilon}(t_{p+1}, t_p) \geq \frac{\Upsilon(t_p)}{1 + K^2 \Upsilon(t_p) \Delta t_p}. \quad (54)$$

This, together with recurrence relation (51), yields

$$1 \leq \overline{\Upsilon}(t_{p+1}, t_p) \Delta t_p (Y + 2K^2). \quad (55)$$

Summing over the first  $r-1$  subintervals then gives

$$\begin{aligned} r-1 &\leq \sum_{p=0}^{r-2} \overline{\Upsilon}(t_{p+1}, t_p) \Delta t_p (Y + 2K^2) \\ &= (t_{r-1} - t_0) \overline{\Upsilon}(t_{r-1}, t_0) (Y + 2K^2). \end{aligned} \quad (56)$$

By solving the above equation for  $r$ , and using the fact that  $(t_{r-1} - t_0) \overline{\Upsilon}(t_{r-1}, t_0) \leq \Delta t \overline{\Upsilon}(t_0 + \Delta t, t_0)$ , we have that

$$r \leq \Delta t \overline{\Upsilon}(t_0 + \Delta t, t_0) (Y + 2K^2) + 1. \quad (57)$$

As  $r$  is an integer, it is also bounded above by

$$r \leq \lceil \Delta t \overline{\Upsilon}(t_0 + \Delta t, t_0) (Y + 2K^2) + 1 \rceil. \quad (58)$$

This also implies that for any  $\gamma > 0$ ,

$$r \leq \lceil \Delta t \overline{\Upsilon}(t_0 + \Delta t, t_0) (Y + 2K^2) + \gamma \rceil. \quad (59)$$

If we take  $\gamma = 1/3$  in (59) and use the definitions of  $A$  and  $Y$ , we find

$$\begin{aligned} r &\leq \left\lceil A^{2k/(2k+1)} r_g^{1/(2k+1)} + 2K^2 \overline{\Upsilon}(t_0 + \Delta t, t_0) \Delta t + 1/3 \right\rceil \\ &= \left\lceil A[1 + (r_g/A - 1)]^{1/(2k+1)} + 2K^2 \overline{\Upsilon}(t_0 + \Delta t, t_0) \Delta t + 1/3 \right\rceil \\ &\leq \left\lceil A + \frac{1}{2k+1} (r_g - A) + 2K^2 \overline{\Upsilon}(t_0 + \Delta t, t_0) \Delta t + 1/3 \right\rceil \\ &\leq \left\lceil A + \frac{1}{3} (3K^2 \overline{\Upsilon}(t_0 + \Delta t, t_0) \Delta t + 2) + 2K^2 \overline{\Upsilon}(t_0 + \Delta t, t_0) \Delta t + 1/3 \right\rceil \\ &\leq \lceil A + 3K^2 \overline{\Upsilon}(t_0 + \Delta t, t_0) \Delta t + 1 \rceil = r_g. \end{aligned} \quad (60)$$

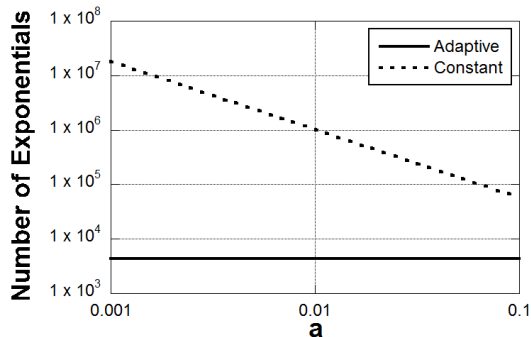


FIG. 1. This figure shows a comparison between the number of exponentials used in our adaptive and constant step size methods for  $H(t) = \exp(-(t-1)^2/a^2)/(a\sqrt{\pi})\mathbb{1}$ ,  $t \in [0, 2]$ ,  $\epsilon = 10^{-4}$  and  $k = 2$ .

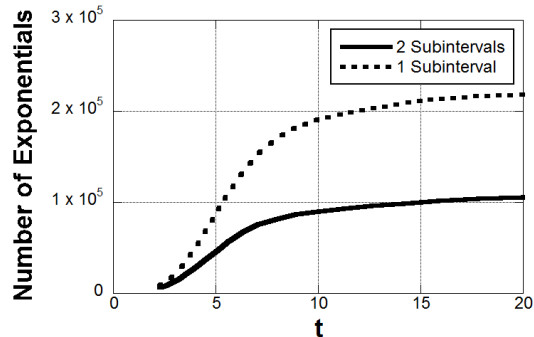


FIG. 2. This plot shows the number of exponentials found by choosing  $k$  adaptively for  $H(t) = t^5 \sin(1/t) \exp(-t)\mathbb{1}$ , and  $\epsilon = 10^{-4}$  for one and two subintervals with  $k = 1$  and  $k = 2$  respectively.

The inequality on the right of Eq. (53) gives

$$\max_{t \in [t_p, t_{p+1}]} \Upsilon(t) \leq \frac{\Upsilon(t_p)}{1 - K^2 \Upsilon(t_p) \Delta t_p}. \quad (61)$$

This, together with the recurrence relation (51), gives

$$\max_{t \in [t_p, t_{p+1}]} \Upsilon(t) \Delta t_p \leq 1/Y. \quad (62)$$

Because  $r \leq r_g$ , the condition (42) of Lemma 7 is satisfied, and the error from the integrator is no more than  $\epsilon/2$ . This implies that the total error is bounded above by  $\epsilon$  if  $n_t$  and  $n_H$  are chosen as in Lemma 1.

We can place an upper bound on the number of exponentials used in the approximation by multiplying  $r$  by the number of exponentials in each  $k^{\text{th}}$ -order Lie-Trotter-Suzuki approximation. We use  $r_g$  as an upper bound for  $r$ , note that there are  $12Md^25^{k-1}$  one-sparse operator exponentials in each of the  $k^{\text{th}}$ -order Lie-Trotter Suzuki approximations, and use  $QueryCost = CN_{\text{exp}}$ , giving Eq. (48). The bound on  $N_T$  follows from the fact that  $QueryCost = CN_{\text{exp}}$ , and  $N_T \leq N_{\text{exp}}/3d^2$ .  $\square$

The result of Theorem 8 shows that adaptive time steps can be used to dramatically reduce the complexity of simulating certain time-dependent Hamiltonian evolutions. This improvement stems from the fact that the performance of the constant time step algorithm depends on the largest possible value of  $\Upsilon$ , whereas the performance of the adaptive version scales with the average value of  $\Upsilon$ . In some cases, such as an example in the following section, the average can be much smaller than the largest value of  $\Upsilon$ , leading to a substantial difference in performance.

## VII. EXAMPLES

We now examine the performance of our adaptive time step decomposition method for a pair of examples. In the first example, we examine a Hamiltonian that is a Gaussian approximation to a Dirac-delta function. In our second example, we examine a non-analytic Hamiltonian, and show that we can substantially reduce the number of exponentials used in some simulations by choosing  $k$  adaptively as well.

The Hamiltonian that we choose in our first example is, for  $a > 0$ ,

$$H(t) = \exp(-(t-1)^2/a^2)/(a\sqrt{\pi})\mathbb{1}. \quad (63)$$

We compare the cost by plotting  $N_{\text{exp}}$  as a function of  $a$  for both methods. We choose our approximations to be 2<sup>nd</sup>-order Lie-Trotter-Suzuki formulae ( $k = 2$ ) with  $\epsilon = 10^{-3}$ . For our adaptive method, rather than choosing the time steps using the upper bound on  $r_g$  as in Theorem 8, we use an iterative process to find the number of steps. That is, we choose each step using (42) with an initial guess of  $r = r_g$ , then find a sequence of steps via (42), and then count the number of steps to find a new guess for  $r$ . Iterating this process gives the solution for  $r$ .

Fig. 1 shows that the adaptive method results in a value of  $N_{\text{exp}}$  that is approximately constant in  $a$ . In comparison, the number of exponentials used in the constant step size case diverges as  $a$  approaches zero. This divergence occurs because  $\lim_{a \rightarrow 0} \max_{t \in [0,2]} \Upsilon_4(t) = \infty$ , whereas the adaptive method yields a nearly constant value of  $N_{\text{exp}}$ , because  $\int_0^2 \Upsilon_4(t) dt$  only weakly depends on  $a$ . This shows that it can be advantageous to choose the step size adaptively if  $H(u)$  varies substantially over the interval of simulation.

The Hamiltonian that we use in our second example is

$$H(t) = t^5 \sin(1/t) \exp(-t) \mathbb{1}. \quad (64)$$

The second-order Lie-Trotter-Suzuki formula should not be used as an approximation to  $U(\Delta t, 0)$  [12], because the third derivative of  $H(t)$  diverges near  $t = 0$ . However,  $U_2$  can be used to approximate the time-evolution on any closed interval that excludes  $t = 0$ . A natural way to handle this is to divide the approximation into two subintervals, and use different values of  $k$  to approximate the evolution within each subinterval. The evolution in the first subinterval,  $u \in [0, t']$ , is approximated using  $U_1$ , whereas the evolution in the remaining subinterval is approximated using  $U_2$ . We reduce the number of exponentials that are used in our simulation schemes for any fixed value of  $t'$  by applying our adaptive step size algorithm in each subinterval. We then vary  $t'$  using a gradient search method to change the size of each subinterval to further reduce the number of exponentials that are used in the simulation.

We show in Fig. 2 that choosing  $k$  adaptively can lead to reductions in the number of exponentials that are needed to approximate the time-evolution operator. Therefore, when simulating the evolution generated by Hamiltonians with singularities, it may be more efficient to use lower-order formulæ near the singularity, and higher-order formulæ further away from it. This method may have applications in situations where high-order integrators fail to provide the scaling expected for singular differential equations, such as those that occur in Coulomb problems [23].

## VIII. SIMULATING ONE-SPARSE HAMILTONIANS

In Sections V and VI we specified the cost of simulating the evolution as a function of  $C$ , which is the number of oracle calls that are needed to simulate a one-sparse Hamiltonian. In this section, we provide an upper bound for  $C$  and discuss how this cost relates to those obtained using other oracle definitions. We then use this bound for  $C$  in concert with our results from Sec. VI to prove Theorem 2 and Corollary 3.

**Lemma 9.** *Let  $\{H_\mu : \mu = 1, \dots, M\}$ , where  $H_\mu : \mathbb{R} \mapsto \mathbb{C}^{2^n \times 2^n}$ , be a set of time-dependent Hermitian operators that is  $2k$ -smooth on  $[t_0, t_0 + \Delta t]$ , and let  $\{H_{\mu,j}\}$  represent the set of one-sparse Hamiltonians that result from applying the BACS decomposition algorithm to each  $\{H_\mu\}$ . The query complexity,  $C$ , of simulating  $\exp\{-iH_{\mu,j}(\tau)\delta t\}$  for any  $j$  and  $[\tau, \tau + \delta t] \subseteq [t_0, t_0 + \Delta t]$  using the oracles **QCol** and **QMatrixVal** and  $n_H$  bits of precision to represent the matrix elements, is bounded above by*

$$C \leq 4n(z_n + 2) + 3n_H, \quad (65)$$

where  $z_n$  is the number of times the mapping  $z \mapsto \lceil 2 \log_2(z) \rceil$  must be iterated, starting at  $n$ , before reaching a value that is at most 6.

To prove this Lemma, we take advantage of the polar decomposition of the matrix elements of the Hamiltonians. This enables a remarkably efficient simulation of the evolution under one-sparse operator exponentials using a sequence of rotations that are controlled by only one qubit at a time. In particular, the result is as in the following Lemma.

**Lemma 10.** *Given that the assumptions of Lemma 9 are met, and if  $n_H$  bits of precision are used to represent each of the matrix elements of  $H_j$ , then  $\exp(-iH_{\mu,j}(\tau)\delta t)$  can be simulated using one qubit to store these matrix elements and  $3n_H$  queries to the oracle **QMatrixVal**.*

*Proof of Lemma 10.* The decomposition scheme of BACS [8] takes a given row number  $x$ , and determines if there will be a matrix element in row  $x$  assigned to  $H_{\mu,j}$ . If there is, then the boolean function  $\xi(x)$  is set to 1; otherwise it is zero. If there is a matrix element, then the column number is determined, and  $M_x$  and  $m_x$  are set equal to the row and column numbers, such that  $M_x \geq m_x$ . The decomposition method also generates a three-bit string  $\nu(x)$ , which we will not otherwise use in the simulation scheme. The BACS decomposition method will therefore transform the initial state  $|\psi\rangle$  according to

$$|\psi\rangle = \sum_x a_x |x, 0^{\otimes 2n+4}\rangle \mapsto \sum_x a_x |x, m_x, M_x, \nu(x), \xi(x)\rangle. \quad (66)$$

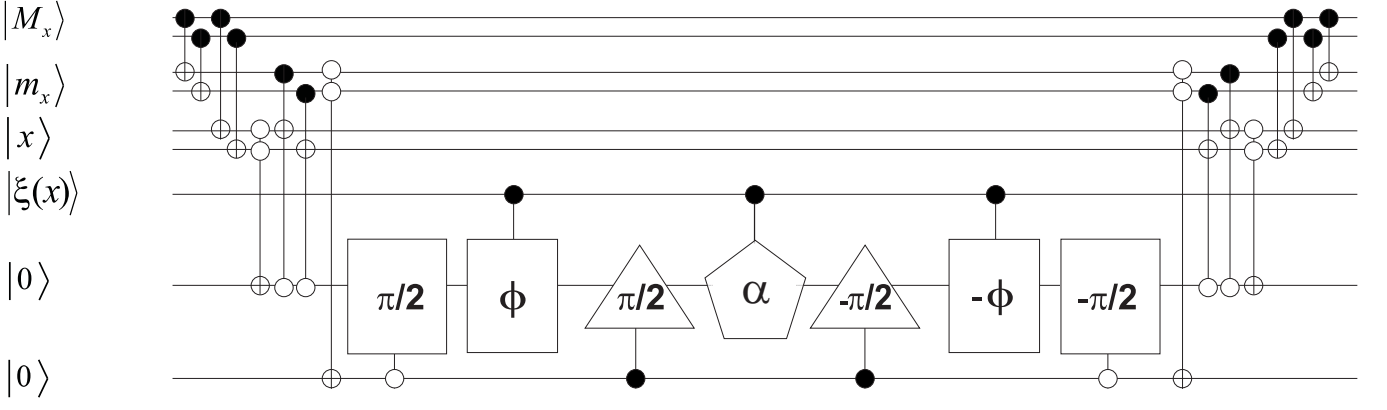


FIG. 3. This circuit simulates  $\exp(-iH_{\mu,j}(t_p)\delta t_p)$  for the one-sparse Hamiltonian  $H_{\mu,j}$ , given an input state of the form of the LHS of (67). Here the variable  $\phi = \text{Arg}([H_{\mu,j}(t_p)]_{m_x, M_x})$  and  $\alpha = 2|[H_{\mu,j}(t_p)]_{m_x, M_x}|\Delta t_p$ . Here we also use rectangles to represent  $R_z$  rotations by a fixed angle, triangles represent  $R_x$  rotations and the pentagon represents a  $R_y$  rotation. These rotations can be enacted by querying the oracle **QMatrixVal** and performing controlled rotations on the output.

Then given this transformed state, our next goal is to perform a mapping between the subspace  $\text{span}\{|m_x\rangle, |M_x\rangle\}$  and an ancilla qubit space. The purpose of this is to map this two-dimensional subspace onto one that can be evolved using single-qubit operations. This transformation is

$$\sum_x a_x |x, m_x, M_x, \nu(m_x), \xi(m_x), 0, 0\rangle \mapsto \sum_x a_x |0^n\rangle \otimes \begin{cases} x = M_x = m_x, & |m_x \oplus M_x, M_x, \nu(m_x), \xi(x), 1, 1\rangle \\ x = M_x \neq m_x, & |m_x \oplus M_x, M_x, \nu(m_x), \xi(x), 1, 0\rangle \\ x = m_x \neq M_x, & |m_x \oplus M_x, M_x, \nu(m_x), \xi(x), 0, 0\rangle \end{cases} \quad (67)$$

The second last qubit is the one that encodes the two-dimensional subspace; it is  $|1\rangle$  if  $x = M_x$ , and  $|0\rangle$  if  $x = m_x$ . The last qubit is used to indicate if  $x$  is a member of a one-dimensional subspace.

Given a state in the form of the RHS of (67), we can simulate the evolution of  $|x\rangle$  by evolving the second last ancilla qubit, whose state is logically equivalent to  $|x\rangle$ . To do so, we must know whether  $x$  is in a one- or two-dimensional irreducible subspace. Specifically, the evolution operator takes one of two possible forms on the subspace  $\text{span}(|m_x\rangle, |M_x\rangle)$ . It performs the transformation

$$a_{M_x} \mapsto a_{M_x} \exp(-i[H_{\mu}(t_p)]_{m_x M_x} \Delta t) \quad (68)$$

or

$$\begin{bmatrix} a_{m_x} \\ a_{M_x} \end{bmatrix} \mapsto \exp\left(-i \begin{bmatrix} 0 & [H_{\mu}(t_p)]_{m_x M_x} \\ ([H_{\mu}(t_p)]_{m_x M_x})^* & 0 \end{bmatrix} \Delta t\right) \begin{bmatrix} a_{m_x} \\ a_{M_x} \end{bmatrix}, \quad (69)$$

if the subspace is one- or two-dimensional, respectively.

We can write the two-dimensional rotation as a sequence of Pauli- $z$  and - $y$  rotations using standard decomposition techniques [15]. Given  $H_{m_x, M_x} = \rho \exp(i\phi)$ , the resulting decomposition of the two-dimensional transformation is

$$\exp\left(-i \begin{bmatrix} 0 & [H_{\mu}(t_p)]_{m_x M_x} \\ [H_{\mu}(t_p)]_{m_x M_x}^* & 0 \end{bmatrix} \Delta t_p\right) = R_z(-\pi/2)R_z(-\phi)R_y(2\rho\Delta t_p)R_z(\phi)R_z(\pi/2). \quad (70)$$

The sequence of exponentials in (70) can be implemented using a sequence of Pauli rotations that are controlled by the qubits that encode the matrix element  $H_{m_x, M_x}$ .

The one-dimensional subspace is evolved according to

$$|x\rangle \rightarrow \exp\left(-i[H_{\mu}(t_p)]_{xx} \Delta t_p\right) |x\rangle. \quad (71)$$

Because we have encoded  $|x\rangle$  in the one-dimensional case to be  $|1\rangle$  in (67), the one-dimensional transformation can be expressed as  $R_z(-2[H_{\mu}(t_p)]_{M_x M_x} \Delta t_p)$ . This rotation can also be written as,

$$R_z(-2[H_{\mu}(t_p)]_{M_x M_x} \Delta t_p) = R_z(-\phi)R_x(-\pi/2)R_y(2\rho\Delta t_p)R_x(\pi/2)R_z(\phi), \quad (72)$$



which allows us to write the one-dimensional rotation in a form that is similar to the two-dimensional rotation.

We present a circuit in Fig. 3 that enacts both the transformation in (67) and also (72) and (70) coherently on each subspace. We combine the rotations for both the one- and two-dimensional cases together in a single sequence of rotations, by making the  $\pi/2$  rotations controlled by the last qubit. In addition, we allow the rotations to be controlled by  $\xi(x)$ , so no rotations are performed if no matrix element has been assigned to row  $x$  of  $H_{\mu,j}$ .

The rotations  $R_y(\alpha)$ , where  $\alpha = 2\rho\Delta t_p$ , and  $R_z(\phi)$  in Fig. 3 can each be implemented by calling the oracle **QMatrixVal**  $n_H/2$  times, provided  $n_H$  is even, and equal numbers of bits are used to encode the modulus and phase of the matrix element. Since there are three rotations of this form,  $3n_H/2$  oracle calls can be used to enact them. However, in order to re-use the ancilla bits that record these values, we need to call the oracle another  $3n_H/2$  times. Therefore, the total number of calls made to **QMatrixVal** is bounded above by  $3n_H$ .  $\square$

Note that, in Fig. 3 we can access each qubit of the oracles independently, without needing to store the other qubits. This is because, for each controlled operation (such as  $R_z(\phi)$ ), we can call the oracle for one qubit of precision, perform the rotation for that qubit, then call the oracle again to erase the value, before calling the oracle for the next qubit. Now that we have proven this Lemma, the proof of Lemma 9 is simple.

*Proof of Lemma 9.* The one-sparse matrix exponentials may be performed by using the BACS decomposition technique, then performing the rotations as described in the proof of Lemma 10, then inverting the BACS decomposition technique to restore the ancilla qubits to their original states. The BACS decomposition technique uses  $2(z_n + 2)$  queries to their oracle to identify the irreducible subspace that a basis state is in, and store this information in a qubit string [8]. Using an oracle that only provides one qubit at a time, the number of oracle calls is multiplied by a factor of  $n$ . Another factor of 2 is obtained because the BACS decomposition technique is inverted, yielding a total number of queries of  $4n(z_n + 2)$ . Using Lemma 10, the rotations can be performed using  $3n_H$  calls to **QMatrixVal**, so the total number of oracle calls needed is bounded as in Eq. (65).  $\square$

Using Lemma 9, it is now straightforward to prove Theorem 2, which is our main result in the paper.

*Proof of Theorem 2.* The proof follows directly by substituting the result of Lemma 9 into those of Theorem 8, while noting that the second and third term in (48) are asymptotically subdominant.  $\square$

As discussed in Ref. [12] for the case of constant time steps, if the Hamiltonian is sufficiently smooth then we can choose  $k$  to increase with  $\Delta t$ , so that the complexity scales close to linearly in  $\Lambda\Delta t$ . Here we obtain a similar result for the case where the time steps are chosen adaptively. Theorem 2 provides a guide to choose an optimal value of  $k$ , which then enables us to prove Corollary 3.

*Proof of Corollary 3.* As  $\{H_\mu\}$  is  $\Lambda$ - $\infty$ -smooth, we can choose  $k$  to be any positive integer: in particular we choose  $k = k_0$  where

$$k_0 = \left\lceil \sqrt{\frac{1}{2} \log_{25/3} \left( \frac{d^2 \tilde{\Upsilon}(t_0 + \Delta t, t_0) \Delta t}{\epsilon} \right)} \right\rceil. \quad (73)$$

Equation (73) implies that  $k_0 \in (d^2 \tilde{\Upsilon}(t_0 + \Delta t, t_0) \Delta t / \epsilon)^{o(1)}$  and  $(25/3)^{k_0} \in (d^2 \tilde{\Upsilon}(t_0 + \Delta t, t_0) \Delta t / \epsilon)^{o(1)}$  because the square-root of a logarithm grows slower than a logarithm. We also have

$$(d^2 \tilde{\Upsilon}(t_0 + \Delta t, t_0) \Delta t / \epsilon)^{1/2k_0} \in (d^2 \tilde{\Upsilon}(t_0 + \Delta t, t_0) \Delta t / \epsilon)^{o(1)}, \quad (74)$$

because  $k_0$  increases with  $\tilde{\Upsilon}(t_0 + \Delta t, t_0) \Delta t / \epsilon$ . We then obtain the scaling given in Eq. (19) by substituting these expressions into (48) and using Lemma 9.  $\square$

For Corollary 3 it was required that there exists  $K > 0$  such that  $\Upsilon'(t) \leq K^2 \Upsilon^2(t)$ , because this was part of the conditions of Theorem 2. However, it can be expected that this requirement is automatically satisfied when  $\{H_\mu\}$  is  $\Upsilon$ - $\infty$ -pointwise-smooth, provided  $\Upsilon$  is chosen appropriately (see Appendix B).

Before concluding, we should also estimate the space-complexity of our algorithm. It follows from an analysis of the BACS decomposition algorithm, that the number of qubits needed for our simulation is  $O(n(\log^* n)^2)$ . Unlike the BACS simulation algorithm [8], this number does not depend on  $\|H\|$  and  $\epsilon$ .

## IX. CONCLUSIONS

We introduce in this paper a pair of quantum algorithms that can be used to simulate time-dependent Hamiltonian evolution on quantum computers, using constant-size or adaptively chosen time steps. The adaptive time step method can provide superior performance, but requires that upper bounds on the norm of the Hamiltonian and its derivatives are known throughout the simulation. In both cases, these simulation algorithms can be performed with similar query complexity to the BACS simulation algorithm [8] if  $H(t)$  is a sum of sufficiently smooth terms.

We also show how to resolve pathological examples, such as our earlier example [12] wherein the higher-order derivatives of  $H(t)$  diverge at one point, although this method cannot be used to attain near-linear scaling unless  $H(t)$  is a sum of terms that are piecewise sufficiently smooth. Furthermore, we have shown that the number of operations used in a simulation of time-dependent Hamiltonian evolution can be reduced by using lower-order Lie-Trotter-Suzuki formulæ to approximate time-evolution near singularities in the Hamiltonian, and higher-order formulæ farther away from the singularities. This approach may also be useful in approximating the solutions to singular differential equations on classical computers.

It may be difficult to compute some of these quantities, such as  $\Lambda$ , for some Hamiltonians. In such circumstances our simulation schemes can still be used, but the output state of the simulation may not be correct within an error tolerance of  $\epsilon$ . We recommend that heuristic testing be used to estimate whether the error is within this tolerance in such circumstances. Such a method could involve performing the swap test [24] between the output states of two separate simulations that employ distinct values of the uncertain parameter. However values such as  $n$  and the upper bound for  $\|H_\mu\|$  that the oracle uses to encode the matrix elements must be known in order to perform the simulation.

We quantify the computational complexity by the number of calls to the oracles **QCol**, **QMatrixVal**, and  $\{T_\mu\}$ . These oracles would not typically be fundamental operations, but would be quantum subroutines consisting of sequences of fundamental quantum operations. The oracles can be implemented *efficiently* by the quantum computer if each  $H_\mu(t)$  is row-computable at every time during the simulation, and there are efficient quantum circuits for the basis transformations  $T_\mu$ .

This work leaves open several interesting avenues for investigation. One such avenue is to address the question of whether or not strictly linear-time quantum simulation algorithms are possible if the Hamiltonian is time-dependent and sufficiently smooth. In addition, it would be interesting to determine whether or not it is possible to devise an algorithm for which the query complexity scales poly-logarithmically with the reciprocal of the error tolerance, rather than sub-polynomially as our algorithm does.

### Appendix A: Proof of Lemma 1

In this section we present a proof of Lemma 1. It provides values for both  $n_H$  and  $n_t$  that ensure that the discretization error is bounded above by  $\epsilon/2$ .

*Proof of Lemma 1.* Here we define  $\sigma := \Delta t/2^{n_t}$ , and define  $\tilde{H}(t)$  to be an approximation to the Hamiltonian  $H(t)$  that uses  $n_H$  bit approximations to the matrix elements. We use  $\tilde{H}_{\mu,j}(t)$  to represent an approximation to the Hamiltonian  $H_{\mu,j}$  that is formed by taking  $n_H$  bit approximations to each matrix element of the one-sparse  $H_{\mu,j}$ . Similarly, we define  $\tilde{\tau}$  to be the closest mesh point to the time  $\tau$  that is still inside the interval  $\mathcal{I}$ . In most cases, the nearest mesh point will be at most a distance of  $\sigma/2$  away from  $\tau$ , but there are cases where the distance can be up to  $\sigma$ . This occurs when  $\tau$  is near the boundary of one of the subintervals of  $\mathcal{I}$ . Because the subintervals can have length no shorter than  $\sigma$ , there will always be a mesh point within the subinterval, and the distance will not be greater than  $\sigma$ . This implies that  $|\tilde{\tau} - \tau| < \sigma$ . Then, using this notation, our goal in this proof is to show that values of  $n_t$  and  $n_H$  satisfying the inequalities in (14) guarantee

$$\left\| \prod_{p=1}^{N_{\text{exp}}} T_{\mu_p}^\dagger \exp[-iH_{\mu_p,j_p}(\tau_p)\Delta t_p] T_{\mu_p} - \prod_{p=1}^{N_{\text{exp}}} T_{\mu_p}^\dagger \exp[-i\tilde{H}_{\mu_p,j_p}(\tilde{\tau}_p)\Delta t_p] T_{\mu_p} \right\| \leq \epsilon/2, \quad (\text{A1})$$

where  $H_{\mu_p,j_p}$  denotes an element from a particular sequence of one-sparse Hamiltonians.

Using Eq. (38) we find that, for Hermitian operators  $A$  and  $B$ ,

$$\begin{aligned} \|e^{-iA} - e^{-iB}\| &= \lim_{n \rightarrow \infty} \|\exp(-iA/n)^n - \exp(-iB/n)^n\| \\ &\leq \lim_{n \rightarrow \infty} n \|\exp(-iA/n) - \exp(-iB/n)\| \\ &= \lim_{n \rightarrow \infty} [\|A - B\| + O(1/n)] = \|A - B\|. \end{aligned} \quad (\text{A2})$$

Using this result with the Hermitian operators  $H_{\mu_p, j_p}(\tau_p)$  and  $\tilde{H}_{\mu_p, j_p}(\tilde{\tau}_p)$  gives

$$\left\| \exp[-iH_{\mu_p, j_p}(\tau_p)\Delta t_p] - \exp[-i\tilde{H}_{\mu_p, j_p}(\tilde{\tau}_p)\Delta t_p] \right\| \leq \|H_{\mu_p, j_p}(\tau_p) - \tilde{H}_{\mu_p, j_p}(\tilde{\tau}_p)\| \Delta t_p. \quad (\text{A3})$$

Then we obtain, using Eq. (38),

$$\begin{aligned} & \left\| \prod_{p=1}^{N_{\text{exp}}} T_{\mu_p}^\dagger \exp[-iH_{\mu_p, j_p}(\tau_p)\Delta t_p] T_{\mu_p} - \prod_{p=1}^{N_{\text{exp}}} T_{\mu_p}^\dagger \exp[-i\tilde{H}_{\mu_p, j_p}(\tilde{\tau}_p)\Delta t_p] T_{\mu_p} \right\| \\ & \leq \sum_{p=1}^{N_{\text{exp}}} \|H_{\mu_p, j_p}(\tau_p) - \tilde{H}_{\mu_p, j_p}(\tilde{\tau}_p)\| \Delta t_p \leq \max_p \left\{ \|H_{\mu_p, j_p}(\tau_p) - \tilde{H}_{\mu_p, j_p}(\tilde{\tau}_p)\| \right\} \sum_{p=1}^{N_{\text{exp}}} |\Delta t_p|. \end{aligned} \quad (\text{A4})$$

Our next step is to bound the sum  $\sum_{p=1}^{N_{\text{exp}}} |\Delta t_p|$ . To do so we note that, for the simulation, the time is broken up into  $r$  short intervals, and on each of these the Lie-Trotter-Suzuki formula  $U_k$  is used. We denote the  $r$  intervals by  $\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_r$ , and the durations of these intervals by  $T_1, T_2, \dots, T_r$ . Using Eq. (A.3) of Ref. 12, if the  $p^{\text{th}}$  exponential is part of the  $q^{\text{th}}$  interval, then the duration of that exponential,  $\Delta t_p$ , is at most

$$|\Delta t_p| \leq (2k/3^k)T_q. \quad (\text{A5})$$

The Lie-Trotter-Suzuki formula  $U_k$  is composed of  $12Md^25^{k-1}$  exponentials, each with a duration that is bounded above by (A5). In addition,  $\sum_{q=1}^r T_q \leq \Delta t$ , so the total duration of the exponentials used to simulate the evolution in the interval  $\mathcal{I}_q$  is at most

$$\sum_{p: \tau_p \in \mathcal{I}_q} |\Delta t_p| \leq 8kMd^2(5/3)^{k-1}T_q \leq 8kMd^2(5/3)^{k-1}\Delta t. \quad (\text{A6})$$

Lemma 1 can be used generally in this work, because this relation does not require that the  $r$  intervals have the same duration, or that  $\mathcal{I}$  is a continuous time interval. The only requirement we have used is that the Lie-Trotter-Suzuki integrator  $U_k$  has been used. This is to obtain the relation (A5) and the number of exponentials in the integrator.

Next, using the triangle inequality, we have

$$\left\| H_{\mu_p, j_p}(\tau_p) - \tilde{H}_{\mu_p, j_p}(\tilde{\tau}_p) \right\| \leq \left\| H_{\mu_p, j_p}(\tau_p) - H_{\mu_p, j_p}(\tilde{\tau}_p) \right\| + \left\| H_{\mu_p, j_p}(\tilde{\tau}_p) - \tilde{H}_{\mu_p, j_p}(\tilde{\tau}_p) \right\|. \quad (\text{A7})$$

Using Taylor's theorem, we find that an upper bound for the error due to the time discretization is

$$\left\| H_{\mu_p, j_p}(\tau_p) - H_{\mu_p, j_p}(\tilde{\tau}_p) \right\| \leq \max_{t, \mu} \|\partial_t H_\mu(t)\| \sigma. \quad (\text{A8})$$

By using a value of  $n_t$  that satisfies (14), we obtain

$$\left\| H_{\mu_p, j_p}(\tau_p) - H_{\mu_p, j_p}(\tilde{\tau}_p) \right\| \leq \frac{\epsilon}{(32kMd^2)(5/3)^{k-1}\Delta t}. \quad (\text{A9})$$

Next we consider the error due to the discretization of  $H$ . The matrix elements are encoded in polar form, so errors emerge because of inaccuracies in the modulus as well as the phase. Using the triangle inequality, the error is bounded above by  $\epsilon_\rho + \epsilon_\phi \Lambda_{\text{max}}$ , where  $\epsilon_\phi$  is the discretization error in the phase,  $\epsilon_\rho$  is the error in the modulus, and  $\Lambda_{\text{max}}$  is an upper bound for the magnitudes of the matrix elements. We choose the same number of bits to encode the modulus and the phase. Then taking  $n_H$  to exceed the value in (14), the error in the modulus and phase satisfy

$$\epsilon_\rho \leq \Lambda_{\text{max}} / \left(2^{n_H/2}\right) \leq \frac{1}{8} \frac{\epsilon}{(32kMd^2)(5/3)^{k-1}\Delta t} \quad (\text{A10})$$

$$\epsilon_\phi \leq 2\pi / \left(2^{n_H/2}\right) \leq \frac{1}{8} \frac{2\pi\epsilon}{(32kMd^2)(5/3)^{k-1}\Lambda_{\text{max}}\Delta t} \quad (\text{A11})$$

Using these relations, we obtain

$$\left\| H_{\mu_p, j_p}(\tilde{\tau}_p) - \tilde{H}_{\mu_p, j_p}(\tilde{\tau}_p) \right\| \leq \epsilon_\rho + \epsilon_\phi \Lambda_{\text{max}} \leq \frac{\epsilon}{(32kMd^2)(5/3)^{k-1}\Delta t}. \quad (\text{A12})$$

Inserting Eqs. (A6) and (A7) into Eq. (A4) gives the error bound as

$$\max_p \left\{ \left\| H_{\mu_p, j_p}(\tau_p) - \tilde{H}_{\mu_p, j_p}(\tilde{\tau}_p) \right\| \right\} \sum_{p=1}^{N_{\text{exp}}} |\Delta t_p| \leq \frac{2\epsilon}{(32kMd^2)(5/3)^{k-1}\Delta t} \times 8kMd^2(5/3)^{k-1}\Delta t = \epsilon/2. \quad (\text{A13})$$

□

## Appendix B: Bounds On Derivatives of $\Upsilon$

In our adaptive simulation method we have required that  $\Upsilon$  is chosen such that there exists a constant  $K$  such that  $|\partial_t \Upsilon(t)| \leq K^2 [\Upsilon(t)]^2$  for all times in the interval. We show in this appendix that this requirement is natural by demonstrating that it naturally emerges for Hamiltonians where  $\Upsilon$  is chosen to be the smallest permissible function.

Now define the set of functions  $\{\Upsilon_P\}$  to be the smallest possible functions such that  $\{H_\mu\}$  is  $\Upsilon_P$ - $P$ -pointwise-smooth. Taking the derivative of  $\Upsilon_P(t)$  gives

$$\begin{aligned} \Upsilon'_P(t) &= \lim_{\delta t \rightarrow 0} \frac{\Upsilon_P(t + \delta t) - \Upsilon_P(t)}{\delta t} \\ &\leq \sup \left\{ (\partial_u \|H(u)\|)|_{u=t}, \dots, \left( \partial_u \|\partial_u^{p-1} H(u)\|^{1/p} \right)|_{u=t}, \dots, \left( \partial_u \|\partial_u^P H(u)\|^{1/(P+1)} \right)|_{u=t} \right\} \\ &\leq \sup \left\{ \frac{1}{p} \|H^{(p-1)}(t)\|^{1/p-1} \|H^{(p)}(t)\| : p = 1, \dots, P+1 \right\}. \end{aligned} \quad (\text{B1})$$

There are now two possible cases, either

$$\|H^{(p-1)}(u)\|^{1/p} \leq \|H^{(p)}(u)\|^{1/(p+1)}$$

or

$$\|H^{(p-1)}(u)\|^{1/p} \geq \|H^{(p)}(u)\|^{1/(p+1)}.$$

In the first case we obtain

$$\frac{1}{p} \|H^{(p-1)}(t)\|^{1/p-1} \|H^{(p)}(t)\| \leq \frac{1}{p} \|H^{(p)}(t)\|^{2/(p+1)}, \quad (\text{B2})$$

and in the second case

$$\frac{1}{p} \|H^{(p-1)}(t)\|^{1/p-1} \|H^{(p)}(t)\| \leq \frac{1}{p} \|H^{(p-1)}(t)\|^{2/p}. \quad (\text{B3})$$

From the definition of  $\Upsilon_P(t)$ ,  $\|H^{(p)}(t)\|^{1/(p+1)} \leq \Upsilon_P(t)$  for  $p = 0, 1, \dots, P$ . However, because  $\|H^{(P+1)}(t)\|^{1/(P+2)}$  is not included in the definition of  $\Upsilon_P(t)$ , we use  $\|H^{(P+1)}(t)\|^{1/(P+2)} \leq \Upsilon_{P+1}(t)$  to bound it. Then, using the fact that  $\Upsilon_{P+1}(t) \geq \Upsilon_P(t)$  and  $p \geq 1$ , the derivative of  $\Upsilon_P(t)$  is bounded above by

$$\Upsilon'_P(t) \leq [\Upsilon_{P+1}(t)]^2. \quad (\text{B4})$$

A lower bound for the derivative can be obtained in the same way, giving the general result

$$|\Upsilon'_P(t)| \leq [\Upsilon_{P+1}(t)]^2. \quad (\text{B5})$$

If there exists a constant  $K$  such that for all  $t \in [t_0, t_0 + \Delta t]$ ,  $\Upsilon_{P+1}(t) \leq K \Upsilon_P(t)$ , then we obtain the restriction in Theorem 2,  $|\Upsilon'_P(t)| \leq K^2 [\Upsilon_P(t)]^2$ .

In the case where  $\Upsilon_\infty$  is taken to be the smallest possible function such that  $\{H_\mu\}$  is  $\Upsilon_\infty$ - $\infty$ -pointwise-smooth, this restriction need not be made. We see from taking the limit as  $P \rightarrow \infty$  of (B5) that

$$|\Upsilon'_\infty(t)| \leq [\Upsilon_\infty(t)]^2. \quad (\text{B6})$$

This means that, if  $\{H_\mu\}$  is  $\Upsilon_\infty$ -pointwise-smooth, the condition  $|\Upsilon'(t)| \leq [\Upsilon(t)]^2$  should hold if  $\Upsilon(t)$  is chosen appropriately. (It does not imply this condition, because  $\Upsilon(t)$  could be chosen poorly.)

## ACKNOWLEDGMENTS

NW thanks A. Hentschel and A. Childs for many helpful comments. We acknowledge MITACS research network, General Dynamics Canada, USARO and iCORE for financial support. PH is a CIFAR Scholar, and BCS is a CIFAR Fellow.

---

[1] R. P. Feynman, *Int. J. Theor. Phys.* **21**, 467–488 (1982).

- [2] S. Lloyd, *Science* **273**, 1073–1078 (1996).
- [3] S. Wiesner, arXiv:quant-ph/9603028.
- [4] C. Zalka, *Forts. der Phys.* **46**, 877–879 (1998); *Proc. R. Soc. Lond. Ser. A* **454**, 313–322 (1998).
- [5] B. M. Boghosian and W. Taylor, *Physica D* **120**, 30–42 (1998).
- [6] D. Aharonov and A. Ta-Shma, *Proc. 35<sup>th</sup> Annual ACM Symposium on Theory of Computing*, pp. 20–29 (2003).
- [7] A. M. Childs, “Quantum Information Processing In Continuous Time”, PhD thesis, Massachusetts Institute of Technology (2004).
- [8] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders, *Comm. Math. Phys.* **270**, 359–371 (2007); *Mathematics of Quantum Computation and Quantum Technology*, G. Chen, L. Kauffman, S. J. Lomonaco, eds., ch. 4 (Taylor & Francis, Oxford, 2007).
- [9] A. M. Childs and R. Kothari, arXiv:1003.3683 (2010) [To appear in Proceedings of TQC 2010].
- [10] J. Roland and N. J. Cerf, *Phys. Rev. A* **68**, 062311 (2003).
- [11] M. Suzuki, *Proc. Japan Acad.* **69**, 161–166 (1993).
- [12] N. Wiebe, D. Berry, P. Høyer, and B. C. Sanders, *J. Phys. A: Math. Theor.* **43**, 065203 (2010).
- [13] G. Strang, *SIAM J. Numer. Anal.* **5**, 506 (1968).
- [14] M. Suzuki, *Phys. Lett. A* **146**, 319–323 (1990).
- [15] M. A. Nielsen and I. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).
- [16] D. A. Lidar and H. Wang, *Phys. Rev. E* **59**, 2429–2438 (1999).
- [17] I. Kassal, S. P. Jordan, P. J. Love, M. Mohseni, and A. Aspuru-Guzik, *Proc. Natl. Acad. Sci.* **105**, 18681–18686 (2008).
- [18] I. Kassal, J. D. Whitfield, A. Perdomo-Ortiz, M.-H. Yung and A. Aspuru-Guzik, arXiv:1007.2648 (2010).
- [19] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Guttman and D. A. Spielman, *Proc. 35<sup>th</sup> Annual ACM Symposium on Theory of Computing*, pp 59–68 (2003).
- [20] A. M. Childs, *Comm. Math. Phys.* **294**, 581–603 (2009).
- [21] D. W. Berry and A. M. Childs, arXiv:0910.4157 (2009).
- [22] A. M. Childs and R. Kothari, *Quant. Inf. Comp.* **10**, 669–684 (2010).
- [23] S. A. Chin, *J. Chem. Phys.* **124**, 054106 (2006).
- [24] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, *Phys. Rev. Lett.* **87**, 167902 (2001).
- [25] There is a typographical error in Definition 4 of Ref. [12]. The definition should also allow  $p = 0$ .