


Circuit implementation of oracles used in a quantum algorithm for solving nonlinear partial differential equations

Frank Gaitan ^{*}*Laboratory for Physical Sciences, 8050 Greenmead Drive, College Park, Maryland 20740, USA*

(Received 2 October 2023; accepted 16 February 2024; published 6 March 2024)

In 2020 and 2021, the author introduced quantum algorithms for solving the Navier-Stokes equations and systems of nonlinear partial differential equations (PDEs), respectively. These algorithms make use of three quantum oracles. In this paper, we show how all three oracles can be implemented as quantum circuits. We cost the circuit implementations, determining their depth, width, and number of non-Clifford gates used as a function of user-specified (i) error tolerances, and (ii) algorithm success probability. With these quantum oracle circuits in hand, the quantum Navier-Stokes and PDE algorithms are now completely specified as quantum circuits.

DOI: [10.1103/PhysRevA.109.032604](https://doi.org/10.1103/PhysRevA.109.032604)

I. INTRODUCTION

More than 40 years ago, Feynman [1] argued that a quantum computer (if one could be built) would be capable of efficiently simulating the dynamics of a quantum system. This is in stark contrast to the situation with a classical computer for which such a simulation is an intractable problem, requiring computational resources that grow exponentially with the size of the quantum system. Following Feynman's seminal work, much subsequent research on quantum simulation has focused on quantum systems as the simulation target.

Recent work [2,3] has pointed to a large new application area for quantum computing: quantum simulation of classical nonlinear continuum systems and fields. Simulating such systems is also difficult for a classical computer, as it requires solving systems of nonlinear partial differential equations (PDEs). Important examples with substantial scientific and economic interest are viscous fluids, plasmas, and nonlinear optical media. Their simulation is the central task of the trillion-dollar aerospace industry, the design of fusion reactors, and determination of the performance of fiber-optics communication systems, respectively. References [2] and [3] showed that a quadratic speed-up is possible for these quantum simulations.

Our focus in this paper will be on the quantum algorithms introduced in Refs. [2,3]. The first finds solutions of the Navier-Stokes equations which govern the motion of viscous fluids [2], and the second, more generally, finds solutions of systems of nonlinear PDEs [3]. We note that other quantum algorithms for fluid dynamics [4–13] have been introduced, though we will not consider them here. The algorithms in Refs. [2,3] make use of three quantum oracles. In the following, we show how these oracles can be implemented as quantum circuits.

To place these two quantum algorithms in context, we note that they are the most recent in a line of extensions

of the quantum phase estimation algorithm (QPEA) introduced by Kitaev [14] in 1995. In 2000, Brassard *et al.* [15] introduced the quantum amplitude estimation algorithm (QAEA), whose computational engine is the QPEA, and they showed how it could be used to determine the average value of a Boolean function. In 2001 and 2002, Novak [16] and Heinrichs [17], respectively, showed how the QAEA could be used to find the average of a real-valued function, making possible a quantum integration algorithm that approximates the value of a definite integral. In 2006, Kacewicz [18] showed how these quantum integration algorithms could be used as the basis for a quantum algorithm for solving systems of nonlinear ordinary differential equations (ODEs). Finally, in 2020 and 2021, Refs. [2,3] showed how Kacewicz' quantum ODE algorithm could be promoted to a quantum algorithm for solving the Navier-Stokes equations, and systems of nonlinear PDEs, respectively. We see that over a period of 26 years, the QPEA, originally introduced to factor integers, has provided the computational basis for a sequence of quantum algorithm extensions, each building on its predecessor, that has made it possible to solve a progressively larger family of important applications.

The structure of this paper is as follows. In Sec. II we briefly review the construction of the quantum PDE (QPDE) algorithm. We shall see that it only uses a quantum computer to approximate the value of definite integrals that arise as part of its construction. We also briefly review Novak's quantum integration algorithm and the QAEA. This will allow us to identify the three oracles used by the quantum PDE algorithm, denoted O , Q , and $\Lambda(Q)$, and to specify their action. It is these actions that the oracle circuits must implement. In Secs. III, IV, and V we present the oracle circuits for O , Q , and $\Lambda(Q)$, respectively. For each circuit, we show that it implements the corresponding oracle action given in Sec. II, and we determine its depth, width, and number of non-Clifford gates used as a function of the user-specified error tolerances and algorithm success probability. We stress that our goal in this paper is to obtain circuit implementations for O , Q , and $\Lambda(Q)$. We leave to future work the tasks of making these implementations

*fgaitan@lps.umd.edu

fault-tolerant and optimal. Finally, in Sec. VI we make closing remarks, and in the Appendix we show how a multi-controlled NOT gate can be implemented as a quantum circuit.

II. PRELIMINARIES

In this section, we summarize the construction of the QPDE algorithm [2,3]. We describe the algorithm's construction in Sec. II A, and we will see that it only uses a quantum computer to evaluate definite integrals that arise in its construction. Novak's [16] quantum integration algorithm (QIA) is used to evaluate these integrals, which we briefly describe in Sec. II B. We shall see that it uses the QAEA to return an approximate value for each integral. The QAEA is described in Sec. II C, and we shall see that it, in turn, uses the QPEA to determine the approximate value of the integral. As we proceed through this section, we identify (i) the quantum oracles O , Q , and $\Lambda(Q)$ used in the QIA and QAEA; and (ii) their action on quantum states. It is this action that will be implemented by the oracle circuits presented in Secs. III, IV, and V. Throughout this section, our goal is to present the QPDE algorithm with sufficient detail to allow us to identify the quantum oracles used. Other aspects of the algorithm such as quantum speed-up can be found in Refs. [2,3].

A. QPDE algorithm

The task for the QPDE algorithm is to find an approximate solution of a system of nonlinear PDEs,

$$\frac{\partial \mathbf{U}}{\partial t} = \mathbf{F}[\mathbf{U}, \mathbf{U}_i, \dots, \mathbf{U}_{i_1, \dots, i_n}], \quad (1)$$

where the exact solution $\mathbf{U}(\mathbf{x}, t)$ is a d -component vector field defined on (i) a spatial region D with boundary ∂D ; and (ii) a time interval $0 \leq t \leq T$. The driver function \mathbf{F} is assumed to be nonlinear, and depends on \mathbf{U} and its spatial partial derivatives up to order n , where $\mathbf{U}_i = \partial \mathbf{U} / \partial x_i$; ...; $\mathbf{U}_{i_1, \dots, i_n} = \partial^n \mathbf{U} / \partial x_{i_1} \dots \partial x_{i_n}$. The solution satisfies (i) the initial condition $\mathbf{U}(\mathbf{x}, 0) = \mathbf{g}(\mathbf{x})$, and (ii) a suitable set of boundary conditions that might be of, say, Dirichlet, Neumann, or Robin type. In Sec. SI-2 of the supporting information for Ref. [3], we showed how systems of nonlinear PDEs containing time partial derivatives of order greater than 1 and/or a nonautonomous driver function $\mathbf{F}[t, \mathbf{U}, \mathbf{U}_i, \dots, \mathbf{U}_{i_1, \dots, i_n}]$ could be reduced to the form given in Eq. (1). The QPDE algorithm is thus applicable to a large family of systems of nonlinear PDEs, which includes those typically encountered in science and engineering applications.

At the coarsest level of description, the QPDE algorithm consists of two steps.

Step 1. Replace the spatial continuum by a spatial grid, while leaving the time t a continuous parameter. This reduces the uncountable number of degrees of freedom of the PDE solution at each time t to a finite number. Many methods are available to implement the spatial discretization (finite difference, finite element, spectral, finite volume, etc.). The result of the spatial discretization is to reduce the system of nonlinear PDEs to a system of nonlinear ODEs,

$$\frac{d}{dt} \mathbf{U}(\mathbf{I}, t) = \mathbf{f}[\mathbf{U}(\mathbf{I}, t)], \quad (2)$$

where the exact solution \mathbf{U} is now only defined on the spatial grid points \mathbf{I} .

Step 2. Use a quantum algorithm for solving nonlinear ODEs to solve Eq. (2). In Refs. [2] and [3] we used Kacewicz' quantum nonlinear ODE algorithm [18], though any quantum nonlinear ODE algorithm could be used instead. The initial condition for Eq. (2) follows from that of the original system of PDEs, where now $\mathbf{g}(\mathbf{x})$ is only evaluated at the grid points \mathbf{I} . Similarly, the boundary conditions on the solution of Eq. (2) follow from those of the system of PDEs, only now evaluated at the spatial grid points lying on the boundary. The QPDE algorithm returns the approximate solution $\alpha(\mathbf{I}, t)$ found by the quantum ODE algorithm as the approximate solution of the system of nonlinear PDEs.

1. Quantum ODE algorithm

We briefly describe Kacewicz' quantum ODE algorithm [18]. For a more detailed presentation, see Ref. [18] and also Refs. [2,3].

The task is to find an approximate, bounded solution $\alpha(t)$ to the system of ODEs,

$$\frac{d}{dt} \mathbf{U}(t) = \mathbf{f}[\mathbf{U}(t)], \quad (3)$$

over the time interval $0 \leq t \leq T$, and subject to the initial condition

$$\mathbf{U}(0) = \alpha(0) = \mathbf{U}_0. \quad (4)$$

The driver function $\mathbf{f}[\mathbf{U}(t)]$ is assumed to be nonlinear and a Hölder class function (see Sec. II A 2 for further discussion).

The construction of Kacewicz' quantum ODE algorithm breaks up into five steps.

(i) The algorithm begins by partitioning the time interval $0 \leq t \leq T$ into n primary subintervals by introducing $n + 1$ intermediate times $t_0 = 0, \dots, t_i, \dots, t_n = T$, where $t_i = t_0 + ih$, and $h = T/n$. Let $T_i = [t_i, t_{i+1}]$ denote the i th primary subinterval.

(ii) Each primary subinterval T_i is further subdivided into $N_k = n^{k-1}$ secondary subintervals by introducing $N_k + 1$ intermediate times $t_{i,0} = t_i, \dots, t_{i,j}, \dots, t_{i,N_k} = t_{i+1}$, where $t_{i,j} = t_{i,0} + j\bar{h}$ and $\bar{h} = h/N_k = T/n^k$. We denote the j th secondary subinterval in the i th primary subinterval by $T_{i,j} = [t_{i,j}, t_{i,j+1}]$. We explain how the parameters n and k are assigned values in Sec. II A 2.

(iii) Kacewicz introduces n parameters $\{\mathbf{y}_i | 0 \leq i \leq n - 1\}$, where \mathbf{y}_i is associated with the initial time t_i of the i th primary subinterval T_i . The parameter \mathbf{y}_0 is defined to be equal to the ODE initial condition:

$$\mathbf{y}_0 \equiv \mathbf{U}_0. \quad (5)$$

The remaining $\{\mathbf{y}_i | 1 \leq i \leq n - 1\}$ approximate the exact solution at times t_i : $\mathbf{y}_i \approx \mathbf{U}(t_i)$. We explain how these parameters are assigned values in step (v) below.

(iv) Within each secondary subinterval $T_{i,j}$, Taylor's method [19,20] is used to approximate the exact solution using a truncated Taylor series about the initial time $t_{i,j}$ of

the secondary subinterval $T_{i,j}$:

$$\alpha_{i,j}(t) = \alpha_{i,j}(t_{i,j}) + \sum_{v=0}^r \frac{1}{(v+1)!} \left. \frac{d^v \mathbf{f}}{dt^v} \right|_{\alpha_{i,j}(t_{i,j})} (t - t_{i,j})^{v+1} + O(\bar{h}^{r+2}). \quad (6)$$

The parameter r is chosen so that the $O(\bar{h}^{r+2})$ error is sufficiently small. Thus $\alpha_{i,j}(t)$ approximates the exact solution $\mathbf{U}(t)$ in the secondary subinterval $T_{i,j}$. The approximate solution $\alpha_i(t)$ in the i th primary subinterval T_i is defined to be

$$\alpha_i(t) = \alpha_{i,j}(t), \quad (7)$$

when $t \in T_{i,j}$. It is demanded that the approximate solution $\alpha_i(t)$ be continuous throughout T_i . This requires that the approximate solutions for two adjacent secondary subintervals be equal at their common boundary time

$$\alpha_{i,j+1}(t_{i,j+1}) = \alpha_{i,j}(t_{i,j+1}). \quad (8)$$

Finally, it is required that the value of $\alpha_i(t)$ at the initial time t_i of the i th primary subinterval $T_i = [t_i, t_{i+1}]$ be \mathbf{y}_i :

$$\alpha_i(t_i) = \alpha_{i,0}(t_{i,0}) \equiv \mathbf{y}_i. \quad (9)$$

Given \mathbf{y}_i , Eqs. (6), (8), and (9) allow the approximate solution $\alpha_i(t)$ to be constructed throughout the primary subinterval T_i . This follows since Eq. (9) gives the value of $\alpha_{i,0}(t)$ at the initial time $t_{i,0} = t_i$, which allows all the coefficients appearing in Eq. (6) (when $j = 0$) to be evaluated. This then gives $\alpha_{i,0}(t)$ throughout the secondary subinterval $T_{i,0}$. Equation (8) then gives the initial value $\alpha_{i,1}(t_{i,1})$ for $\alpha_{i,1}(t)$. This initial value then allows $\alpha_{i,1}(t)$ to be constructed using Eq. (6), and Eq. (8) then gives the initial value for $\alpha_{i,2}(t)$. Iterating this procedure across all the secondary subintervals $T_{i,j}$ gives $\alpha_i(t)$ throughout T_i as claimed. We see that once the parameters $\{\mathbf{y}_i \mid 0 \leq i \leq n-1\}$ are known, we can determine the approximate solutions for all of the primary subintervals $\{\alpha_i(t) \mid 0 \leq i \leq n-1\}$. They then allow the approximate solution $\alpha(t)$ to the system of ODEs to be determined by defining

$$\alpha(t) = \alpha_i(t) \quad (10)$$

when $t \in T_i$.

(v) At this point in the algorithm only \mathbf{y}_0 is known. To determine the remaining parameters $\{\mathbf{y}_i \mid 1 \leq i \leq n-1\}$, Eq. (3) is integrated over the primary subinterval $T_i = [t_i, t_{i+1}]$, giving

$$\mathbf{U}(t_{i+1}) = \mathbf{U}(t_i) + \int_{t_i}^{t_{i+1}} dt \mathbf{f}[\mathbf{U}(t)]. \quad (11)$$

Kacewicz then adds and subtracts the integral of $\mathbf{f}[\alpha_i(t)]$ to Eq. (11) to obtain

$$\begin{aligned} \mathbf{U}(t_{i+1}) &= \mathbf{U}(t_i) \\ &+ \int_{t_i}^{t_{i+1}} dt \mathbf{f}[\alpha_i(t)] \\ &+ \int_{t_i}^{t_{i+1}} dt \{\mathbf{f}[\mathbf{U}(t)] - \mathbf{f}[\alpha_i(t)]\}, \end{aligned} \quad (12)$$

which is still exact. To obtain a relation between the $\{\mathbf{y}_i\}$ he makes two approximations. First, recalling that $\mathbf{U}(t_i) \approx \mathbf{y}_i$, he replaces $\mathbf{U}(t_{i+1})$ and $\mathbf{U}(t_i)$ with \mathbf{y}_{i+1} and \mathbf{y}_i , respectively, in

Eq. (12). Second, he discards the third term on the right-hand side (RHS) of Eq. (12) since it is small. Eq. (12) becomes

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \int_{t_i}^{t_{i+1}} dt \mathbf{f}[\alpha_i(t)], \quad (13)$$

with $0 \leq i \leq n-1$. Finally, the integral over T_i is broken up into a sum of integrals over secondary subintervals $T_{i,j}$, and he makes the change of integration variable $t = t_{i,j} + \bar{h}\tau$ in each secondary subinterval, giving

$$\mathbf{y}_{i+1} = \mathbf{y}_i + N_k \sum_{j=0}^{N_k-1} \frac{\bar{h}}{N_k} \int_0^1 d\tau \mathbf{f}[\alpha_{i,j}(t(\tau))]. \quad (14)$$

Given \mathbf{y}_i and the integrals over the secondary subintervals, Eq. (14) determines \mathbf{y}_{i+1} . This allows the $\{\mathbf{y}_i\}$ to be determined iteratively. The iteration begins with the primary subinterval T_0 where we have $\mathbf{y}_0 = \mathbf{U}_0$ [see Eq. (5)]. As noted above, knowing \mathbf{y}_0 allows the $\{\alpha_{0,j}(t) \mid 0 \leq j \leq N_k-1\}$ to be determined, and thus $\alpha_0(t)$. This then determines the integrands appearing in Eq. (14) for $i = 0$. Once the integrals are evaluated, Eq. (14) determines \mathbf{y}_1 . We can now repeat this procedure. Knowing \mathbf{y}_1 determines the $\{\alpha_{1,j}(t)\}$, and thus $\alpha_1(t)$. This determines the integrands for Eq. (14) when $i = 1$. Evaluating the integrals and adding the result to \mathbf{y}_1 determines \mathbf{y}_2 . In this way, the algorithm steps through all the primary subintervals. Along the way, the approximate solutions for the primary subintervals $\{\alpha_i(t) \mid 0 \leq i \leq n-1\}$ are determined, which then give the approximate solution $\alpha(t)$ of the system of ODEs through Eq. (10).

Kacewicz uses a quantum integration algorithm to evaluate the integrals over the secondary subintervals. It is important to appreciate that this is the *only* task in Kacewicz' quantum ODE algorithm (and thus also in the QPDE algorithm) that requires a quantum computer. In Sec. II B we focus on Novak's QIA [16], showing how it is constructed, and meeting the first of our quantum oracles.

2. Loose ends

Here we make good on two promises made in the above discussion. (a) We define Hölder class functions, and (b) we explain how the time partitioning parameters n and k are assigned values.

(a) As noted above, we consider nonlinear driver functions $\mathbf{f}[\mathbf{U}(t)] \equiv \mathbf{F}(t)$ that are Hölder class functions. Such a driver function belongs to one of a family of function spaces $\{C^{r,\rho}\}$ parametrized by r and ρ [21]. Elements of $C^{r,\rho}$ are continuously differentiable up to order r , with the r th derivative satisfying the upper bound,

$$\left\| \frac{d^r \mathbf{F}(t)}{dt^r} - \frac{d^r \mathbf{F}(t')}{dt^r} \right\| < C |t - t'|^\rho, \quad (15)$$

where the Hölder exponent ρ satisfies $0 < \rho \leq 1$. Kacewicz showed that for Hölder class driver functions, the error in the approximate solution $\alpha(t)$ returned by his algorithm satisfies

$$\varepsilon = \sup_{0 \leq t \leq T} \|\mathbf{U}(t) - \alpha(t)\| = O(1/n^{\alpha_k}) \quad (16)$$

with probability $1 - \delta$. Here n is the number of primary subintervals; k is related to the number of secondary subintervals in a primary subinterval: $N_k = n^{k-1}$, $\alpha_k = k(q+1) - 1$, and

$q = r + \rho$. To achieve this performance, the error ε_1 in the result returned by the QAEA (see Sec. II C) must satisfy

$$\varepsilon_1 = 1/n^{k-1}, \quad (17)$$

and the QAEA success probability $1 - \delta_1$ must satisfy

$$1 - \delta_1 = (1 - \delta)^{1/n^k}, \quad (18)$$

where δ is the failure probability defined below Eq. (16).

(b) Here we explain how n and k are assigned values. Because Kacewicz' algorithm is an explicit algorithm, the smallest timescale \bar{h} must be less than the Courant-Friedrichs-Lewy (CFL) time Δt_{CFL} if simulation of the algorithm is to be stable [22]. For an example of how Δt_{CFL} is calculated, see Sec. SI-5E in the supplementary information of Ref. [2]. Let N_{tot} be defined by the relation $T = N_{\text{tot}} \Delta t_{\text{CFL}}$, where T is the run-time of the quantum algorithm. In Kacewicz' time partitioning, a total of n^k secondary subintervals of duration \bar{h} are introduced so that $T = \bar{h} n^k$. Equating these two expressions for T gives

$$\frac{\bar{h}}{\Delta t_{\text{CFL}}} = \frac{N_{\text{tot}}}{n^k}. \quad (19)$$

We require $\bar{h}/\Delta t_{\text{CFL}} < 1$ so that the time partition is compatible with the CFL stability criterion. We choose n and k so that this is true. To that end, we solve Eq. (17) for k :

$$k = 1 + \lceil \log_2(1/\varepsilon_1)/\log_2 n \rceil, \quad (20)$$

where $\lceil x \rceil$ is the smallest integer greater than x . We use the following iterative procedure to assign values to n and k . First, we choose values for N_{tot} and ε_1 , and we make an initial guess n_{in} for n . Set $n_0 = n_{\text{in}}$ and use Eq. (20) to determine k_0 using n_0 for n . Then evaluate the ratio $N_{\text{tot}}/n_0^{k_0}$. If the ratio is less than 1, then Eq. (19) gives $\bar{h}/\Delta t_{\text{CFL}} < 1$ and we accept n_0 and k_0 as n and k . Otherwise, increment $n_i = n_{i-1} + 1$ and determine k_i using Eq. (20). Then determine the new ratio $N_{\text{tot}}/n_i^{k_i}$ and repeat this procedure until the current value of $N_{\text{tot}}/n_a^{k_a}$ is less than 1 for the first time. Accept n_a and k_a as n and k and use these values to partition the time interval $[0, T]$ as described in Sec. II A 1.

B. QIA

We saw in Sec. II A that the QPDE algorithm only uses a quantum computer to evaluate definite integrals [see Eq. (14)] of the form

$$\mathcal{I} = \int_0^1 d\tau f(\tau), \quad (21)$$

where we focus on a single component of the integrand $\mathbf{f}[\boldsymbol{\alpha}_{i,j}(t(\tau))]$ in Eq. (14). Novak [16] and Heinrich [17] introduced quantum algorithms that approximate the value of a definite integral such as \mathcal{I} using the QAEA [15]. The latter algorithm requires $0 \leq f(\tau) \leq 1$, which is not true for arbitrary definite integrals for at least two reasons. First, the integration domain $[a, b]$ need not be $[0, 1]$. This is easily rectified by a linear transformation mapping $[a, b] \rightarrow [0, 1]$. Second, $f(\tau)$ may not be restricted to the range $[0, 1]$. This too can be easily dealt with. Since $f(\tau)$ is assumed to be continuous over the integration domain, it has bounded variation and so takes a maximum (minimum) value f_{max} (f_{min}) over this domain. We

introduce a new function $g(\tau)$, which is a shifted and rescaled version of $f(\tau)$:

$$g(\tau) = \frac{f(\tau) - f_{\text{min}}}{\Delta f}, \quad (22)$$

where $\Delta f \equiv f_{\text{max}} - f_{\text{min}}$. By construction, $0 \leq g(\tau) \leq 1$. Solving Eq. (22) for $f(\tau)$ and inserting the result into Eq. (21) gives the relation

$$\mathcal{I} = f_{\text{min}} + \Delta f \mathcal{G}, \quad (23)$$

where

$$\mathcal{G} = \int_0^1 d\tau g(\tau). \quad (24)$$

The definite integral \mathcal{G} has the properties required for application of the QAEA, and hence the QIA. We will use Novak's QIA to determine an approximate value for \mathcal{G} , which then [through Eq. (23)] gives an approximate value for the desired integral \mathcal{I} .

Novak's QIA begins by replacing \mathcal{G} by its Riemann sum \mathcal{G}_R ,

$$\mathcal{G} \rightarrow \mathcal{G}_R = \frac{1}{M_2} \sum_{j=0}^{M_2-1} g(\tau_j), \quad (25)$$

where $g(\tau)$ is evaluated at M_2 uniformly spaced points in the integration domain $[0, 1]$. We write M_2 to match the notation used in Sec. III.

Novak introduced a quantum oracle O that encodes the M_2 values of $g(\tau)$ appearing in the Riemann sum \mathcal{G}_R into the quantum state of two registers. The first register contains $m_2 = \lceil \log_2 M_2 \rceil$ qubits, while the second contains one qubit. The oracle O acts on the initial state $|\psi_0\rangle$:

$$\begin{aligned} |\psi_0\rangle &= (H^{m_2} \otimes X)|0\rangle_{m_2} \otimes |0\rangle_1 \\ &= \frac{1}{\sqrt{M_2}} \sum_{j=0}^{M_2-1} |j\rangle_{m_2} \otimes |1\rangle_1, \end{aligned} \quad (26)$$

where H^{m_2} denotes the Hadamard gate H applied transversally to the qubits in the first register, and X is the one-qubit Pauli X gate. A subscript m on a quantum state indicates that it is an m -qubit state. In the following, this subscript will be omitted when context makes clear its value. The computational basis states (CBS) $\{|j\rangle | 0 \leq j \leq M_2 - 1\}$ for the first register are

$$|j\rangle = |j_0\rangle \otimes \cdots \otimes |j_{m_2-1}\rangle, \quad (27)$$

where $j = \sum_{k=0}^{m_2-1} j_k 2^k$. The action of O on $|\psi_0\rangle$ is defined to be

$$\begin{aligned} |\psi\rangle &= O|\psi_0\rangle \\ &\equiv \frac{1}{\sqrt{M_2}} \sum_{j=0}^{M_2-1} |j\rangle \otimes [\sqrt{g(\tau_j)}|1\rangle + \sqrt{1-g(\tau_j)}|0\rangle]. \end{aligned} \quad (28)$$

Introducing the normalized and mutually orthogonal states $|n_1\rangle$ and $|n_0\rangle$,

$$|n_1\rangle = \frac{1}{\sqrt{M_2}} \sum_{j=0}^{M_2-1} \sqrt{\frac{g(\tau_j)}{\bar{g}}} |j\rangle \otimes |1\rangle, \quad (29)$$

$$|n_0\rangle = \frac{1}{\sqrt{M_2}} \sum_{j=0}^{M_2-1} \sqrt{\frac{1-g(\tau_j)}{1-\bar{g}}} |j\rangle \otimes |0\rangle, \quad (30)$$

allows $|\psi\rangle$ to be written as

$$|\psi\rangle = \sqrt{a}|n_1\rangle + \sqrt{1-a}|n_0\rangle, \quad (31)$$

where

$$\bar{g} = \frac{1}{M_2} \sum_{j=0}^{M_2-1} g(\tau_j), \quad (32)$$

and direct calculation shows that

$$a = \bar{g}. \quad (33)$$

Comparing Eqs. (32) and (25), we see that

$$a = \bar{g} = \mathcal{G}_R. \quad (34)$$

Thus, determining a in Eq. (31) determines the desired approximation for the integral \mathcal{G} , and hence for \mathcal{I} [Eq. (23)]. As we shall see in Sec. II C, given the state $|\psi\rangle$, the QAEA returns an estimate for a , and thus for \mathcal{I} . We discuss the error in this estimate in Sec. II C.

C. QAEA

Here we briefly describe the QAEA. (i) We identify the oracles Q and $\Lambda(Q)$ introduced by the algorithm and present their action on states (Sec. II C 1); and (ii) we relate the error in the estimate returned by Novak's QIA for the value of a definite integral (Sec. II C 2) to the error in the estimate returned by the QAEA for the value of the integral's Riemann sum.

1. Q and $\Lambda(Q)$

The QAEA [15] assumes the Hilbert space \mathcal{H} partitions into two subspaces \mathcal{H}_1 and \mathcal{H}_0 referred to as the ‘‘good’’ and ‘‘bad’’ subspaces, respectively. It also assumes a quantum algorithm \mathcal{A} exists that implements the unitary operator A , whose action on the $|0\rangle$ CBS produces a state $|\psi\rangle$ of the form

$$\begin{aligned} |\psi\rangle &= A|0\rangle \\ &= \sqrt{a}|n_1\rangle + \sqrt{1-a}|n_0\rangle. \end{aligned} \quad (35)$$

Here, (i) $|n_1\rangle$ and $|n_0\rangle$ are normalized, orthogonal states, with $|n_1\rangle$ ($|n_0\rangle$) belonging to the good (bad) subspace \mathcal{H}_1 (\mathcal{H}_0); and (ii) \sqrt{a} is the amplitude that a measurement of $|\psi\rangle$ in the computational basis (CB) will yield a good state. The task of the QAEA is to return an estimate $\sqrt{\tilde{a}}$ of the amplitude \sqrt{a} , and a bound on the error $\delta a = |\tilde{a} - a|$.

To that end, a Grover-like operator Q is introduced whose action on the two-dimensional subspace \mathcal{H}_ψ spanned by $|n_1\rangle$ and $|n_0\rangle$ is defined as

$$Q = U_\psi U_{n_0}, \quad (36)$$

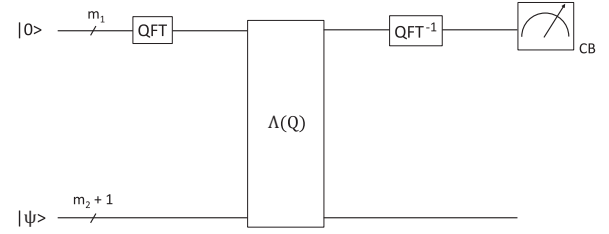


FIG. 1. Quantum circuit for the QPEA. The circuit acts on two registers: the first contains m_1 qubits, and the second contains $m_2 + 1$ qubits. The first register starts in the $|0\rangle$ CB state and the second starts in the state $|\psi\rangle$. The circuit first applies the quantum Fourier transform to the first register, then the operator $\Lambda(Q)$ is applied to both registers. The inverse quantum Fourier transform is next applied to the first register, and finally, the first register is measured in the CB.

where

$$U_\psi = I - 2|\psi\rangle\langle\psi| = A\{I - 2|0\rangle\langle 0|\}A^{-1}, \quad (37a)$$

$$U_{n_0} = I - 2|n_0\rangle\langle n_0|. \quad (37b)$$

Direct calculation shows that Q rotates \mathcal{H}_ψ by an angle $2\theta_a$:

$$Q|n_1\rangle = \cos 2\theta_a|n_1\rangle - \sin 2\theta_a|n_0\rangle, \quad (38a)$$

$$Q|n_0\rangle = \sin 2\theta_a|n_1\rangle + \cos 2\theta_a|n_0\rangle, \quad (38b)$$

where $\sqrt{a} \equiv \sin \theta_a$ and $0 \leq \theta_a \leq \pi/2$. With this definition, the task of estimating \sqrt{a} reduces to estimating the angle θ_a . The QAEA uses the QPEA [14] to estimate θ_a . The circuit for the QPEA appears in Fig. 1.

The QPEA circuit acts on two registers, the first containing m_1 qubits, and the second containing $m_2 + 1$ qubits. We explain how m_2 and m_1 are determined in Secs. III and IV, respectively. The initial state $|\psi\rangle$ for the second register has the form appearing in Eq. (35). The circuit first applies the quantum Fourier transform to the first register; then it applies the operator $\Lambda(Q)$ to both registers; next the inverse quantum Fourier transform is applied to the first register; and finally, the first register is measured in the CB.

The operator $\Lambda(Q)$ has the following action on the two-register CB states:

$$\Lambda(Q)|j\rangle_{m_1} \otimes |k\rangle_{m_2+1} = |j\rangle_{m_1} \otimes Q^j|k\rangle_{m_2+1}. \quad (39)$$

$\Lambda(Q)$ thus applies Q to the second register j times when the first register is in the CB state $|j\rangle$.

The QAEA thus makes use of the two related oracles Q and $\Lambda(Q)$, as well as the oracle A used to prepare the state $|\psi\rangle$ [Eq. (35)]. As seen above, the QPEA algorithm uses Novak's oracle O to prepare the state $|\psi\rangle$ [Eqs. (28) and (31)], which has the form appearing in Eq. (35):

$$\begin{aligned} |\psi\rangle &= O(H^{m_2} \otimes X)|0\rangle_{m_2} \otimes |0\rangle_1 \\ &= \sqrt{a}|n_1\rangle + \sqrt{1-a}|n_0\rangle, \end{aligned} \quad (40)$$

with $|n_1\rangle$ and $|n_0\rangle$ defined in Eqs. (29) and (30), respectively. Thus Novak's QIA prepares the state $|\psi\rangle$ using the operator

$$A = O(H^{m_2} \otimes X). \quad (41)$$

Equations (28), (38)–(39) thus specify the action of the oracles O , Q , and $\Lambda(Q)$, respectively. Quantum circuits implementing these actions appear in Secs. III, IV, and V, respectively.

2. Error bounds

The QAEA provides an upper bound on the error $\delta a = |\tilde{a} - a|$,

$$\delta a \leq \frac{2\pi k}{M_1} \sqrt{a(1-a)} + \left(\frac{k\pi}{M_1}\right)^2, \quad (42)$$

which is satisfied with probability $1 - \delta_1$: (i) of at least $8/\pi^2$ when $k = 1$; and (ii) greater than $1 - 1/\{2(k-1)\}$ for $k \geq 2$. Here $M_1 = 2^{m_1}$ and so the error δa decreases exponentially with the number of qubits m_1 (see Fig. 1). Note that, as shown in Ref. [17], by running the QAEA multiple times, and returning the median of the \tilde{a} values found, the probability $1 - \delta_1$ of satisfying the error bound can be brought arbitrarily close to 1. See Sec. VI for further discussion.

For Novak's QIA we saw that $a = \bar{g} = \mathcal{G}_R$ [Eq. (34)]. If \tilde{g} denotes the estimate returned by the QAEA for \bar{g} , then Eq. (42) gives the error $\delta g \equiv |\tilde{g} - \bar{g}|$:

$$\delta g \leq \frac{2\pi k}{M_1} \sqrt{\bar{g}(1-\bar{g})} + \left(\frac{k\pi}{M_1}\right)^2. \quad (43)$$

For $M_1 \gg 1$ and $k = 1$, this becomes

$$\delta g \leq \frac{2\pi}{M_1}. \quad (44)$$

If the desired error tolerance for δg is denoted ε_1 , then choosing $M_1 > 2\pi/\varepsilon_1$ insures

$$\delta g < \varepsilon_1 \quad (45)$$

and so

$$\delta g = O(\varepsilon_1). \quad (46)$$

As we saw in Sec. II B, the QIA algorithm returns an estimate of the definite integral

$$\mathcal{I} = \int_0^1 d\tau f(\tau). \quad (47)$$

If we replace \mathcal{I} by a Riemann sum \mathcal{I}_R ,

$$\mathcal{I}_R = \frac{1}{M_2} \sum_{j=0}^{M_2-1} f(\tau_j) \equiv \tilde{f}, \quad (48)$$

then we have

$$\mathcal{I} = \mathcal{I}_R + O(1/M_2). \quad (49)$$

The resulting discretization error

$$\varepsilon_d = |\mathcal{I} - \mathcal{I}_R| \quad (50)$$

is thus

$$\varepsilon_d = O(1/M_2). \quad (51)$$

Equation (23) relates \mathcal{I} to the integral \mathcal{G} of the shifted and rescaled function $g(\tau)$ [Eq. (24)],

$$\mathcal{I} = f_{\min} + \Delta f \mathcal{G}, \quad (52)$$

and a similar relation holds for the Riemann sums,

$$\mathcal{I}_R = f_{\min} + \Delta f \mathcal{G}_R. \quad (53)$$

Since $\mathcal{I}_R = \tilde{f}$ [Eq. (48)] and $\bar{g} = \mathcal{G}_R$, Eq. (53) can be rewritten as

$$\tilde{f} = f_{\min} + \Delta f \bar{g}. \quad (54)$$

We define the estimate \tilde{f} for \tilde{f} in terms of the QAEA estimate \tilde{g} as

$$\tilde{f} = f_{\min} + \Delta f \tilde{g}. \quad (55)$$

Let $\delta f = |\tilde{f} - \tilde{f}| = |\tilde{f} - \mathcal{I}_R|$. Then, Eqs. (46), (54), and (55) give

$$\delta f = \Delta f \delta g = O(\varepsilon_1). \quad (56)$$

The error $\delta \mathcal{I} \equiv |\mathcal{I} - \tilde{f}|$ in the estimate of \mathcal{I} is then

$$\delta \mathcal{I} \leq |\mathcal{I} - \mathcal{I}_R| + |\mathcal{I}_R - \tilde{f}| = \varepsilon_d + \delta f. \quad (57)$$

We see that the error $\delta \mathcal{I}$ in the value of the integral \mathcal{I} is the sum of (i) the discretization error ε_d due to replacing \mathcal{I} by a Riemann sum, and (ii) the QAEA error $\delta f = O(\varepsilon_1)$ in the estimate of the Riemann sum:

$$\delta \mathcal{I} = O(\varepsilon_d) + O(\varepsilon_1). \quad (58)$$

As will be seen in Sec. III, Eq. (58) will receive a third contribution due to the necessarily finite resources present in a quantum computer [see Eq. (75)].

III. IMPLEMENTING O

In this section, we present the circuit that implements Novak's oracle O . As we have seen, O 's action is to encode the integrand values $\{g(\tau_j)\}$ (of the integral whose approximate value we desire) into the state of an $m_2 + 1$ qubit register [see Eqs. (24), (25), and (28)]. As we shall see below, it does this using operators R and S , where R is a controlled-rotation operator whose rotation angles produce the above encoding [Eq. (80)]. Once O has acted, the register state is $|\psi\rangle = \sqrt{a}|n_1\rangle + \sqrt{1-a}|n_0\rangle$, where the approximate value of the integral is stored in the squared amplitude $a \equiv \sin^2 \theta_a$. The QPEA is used to determine an approximate value for θ_a , and thus of the desired integral.

The circuit for O is presented in Sec. III A, and that of the operators R and S in Secs. III B and III C, respectively. Section III D pulls together intermediate results from the above sections to determine the resource requirements needed to implement O .

A. Quantum circuit for O

As seen in Sec. II B, Novak's QIA uses the QAEA to obtain an estimate \tilde{g} for the Riemann sum $\mathcal{G}_R = \bar{g}$,

$$\bar{g} = \frac{1}{M_2} \sum_{j=0}^{M_2-1} g(\tau_j), \quad (59)$$

where the $g(\tau_j)$ are known. As seen in Eq. (28), the oracle O encodes the $g(\tau_j)$ into the state $|\psi\rangle$. Since M_2 is the number of function evaluations appearing in Eq. (59), it sets the scale for the computational work required to determine \bar{g} .

Before introducing the circuit for O , some preliminary remarks are in order.

1. Preliminaries

By construction, the values $g(\tau_j)$ satisfy $0 \leq g(\tau_j) \leq 1$. This allows the introduction of angles $\hat{\gamma}(j)$,

$$\cos\left(\frac{\pi}{2}\hat{\gamma}(j)\right) \equiv \sqrt{g(\tau_j)} \quad (0 \leq j \leq M_2 - 1). \quad (60)$$

By definition, $0 \leq \hat{\gamma}(j) \leq 1$, and so it has the binary expansion

$$\hat{\gamma}(j) = \hat{\gamma}_0(j)\left(\frac{1}{2}\right)^1 + \cdots + \hat{\gamma}_{i-1}(j)\left(\frac{1}{2}\right)^i + \cdots \quad (61)$$

The coefficients $\hat{\gamma}_i(j)$ are (i) binary variables (bits) taking values of 0 or 1; and (ii) easily determined from $\hat{\gamma}(j)$.

Since computers, quantum or classical, have finite resources, the binary expansion in Eq. (61) must be truncated to a finite number of terms. Let $\bar{\gamma}(j)$ be the m_3 -bit approximation of $\hat{\gamma}(j)$:

$$\bar{\gamma}(j) \equiv \bar{\gamma}_0(j)\left(\frac{1}{2}\right)^1 + \cdots + \bar{\gamma}_{m_3-1}(j)\left(\frac{1}{2}\right)^{m_3}, \quad (62)$$

where $\bar{\gamma}_i(j) \equiv \hat{\gamma}_i(j)$ for $0 \leq i \leq m_3 - 1$. From Eqs. (61) and (62), the truncation error $\delta\hat{\gamma}(j) \equiv |\hat{\gamma}(j) - \bar{\gamma}(j)|$ satisfies

$$\delta\hat{\gamma}(j) = O(1/2^{m_3+1}). \quad (63)$$

For later purposes, we define $M_3 \equiv 2^{m_3}$ so that

$$\delta\hat{\gamma}(j) = O(1/M_3). \quad (64)$$

When implementing the quantum circuit for O , we must work with $\bar{\gamma}(j)$ (finite resources again) instead of $\hat{\gamma}(j)$. Consequently, we define

$$G(\tau_j) \equiv \cos^2\left(\frac{\pi}{2}\bar{\gamma}(j)\right) \quad (65)$$

and

$$\bar{G} \equiv \frac{1}{M_2} \sum_{j=0}^{M_2-1} G(\tau_j). \quad (66)$$

Because we work with the $\bar{\gamma}(j)$, the oracle circuit for O will produce the state $|\psi\rangle$ in Eq. (28), but with $g(\tau_j) \rightarrow G(\tau_j)$. Following the discussion in Sec. II C, the QAEA will then return an estimate \tilde{G} for \bar{G} with error

$$\delta G = |\tilde{G} - \bar{G}| = O(\varepsilon_1), \quad (67)$$

where $\varepsilon_1 > 2\pi/M_1$, $M_1 = 2^{m_1}$, and m_1 is the width of the first quantum register in Fig. 1. The error δg in the QAEA estimate of the Riemann sum \bar{g} is then

$$\delta g = |\tilde{G} - \bar{g}| \leq |\tilde{G} - \bar{G}| + |\bar{G} - \bar{g}|. \quad (68)$$

Defining $\delta\bar{g} \equiv |\bar{G} - \bar{g}|$, together with Eq. (67), Eq. (68) can be rewritten as

$$\delta g \leq \delta G + \delta\bar{g}. \quad (69)$$

We can determine $\delta\bar{g}$ using Eqs. (59), (60), (65), and (66),

$$\begin{aligned} \delta\bar{g} &= \frac{1}{M_2} \left| \sum_{j=0}^{M_2-1} \left\{ \cos^2\left(\frac{\pi}{2}\bar{\gamma}(j)\right) - \cos^2\left(\frac{\pi}{2}\hat{\gamma}(j)\right) \right\} \right| \\ &\leq \frac{1}{M_2} \sum_{j=0}^{M_2-1} |\sin \pi \bar{\gamma}(j)| \frac{\pi}{2} \delta\hat{\gamma}(j) + O(\delta\hat{\gamma}(j))^2. \end{aligned} \quad (70)$$

Let ε_γ denote the error tolerance for $\delta\hat{\gamma}(j)$. If $M_3 > 1/\varepsilon_\gamma$, then Eq. (64) gives $\delta\hat{\gamma}(j) = O(\varepsilon_\gamma)$, and so, from Eq. (70),

$$\delta\bar{g} = O(\varepsilon_\gamma). \quad (71)$$

The error $\delta\mathcal{G} = |\tilde{G} - \mathcal{G}|$ in the estimate of the integral \mathcal{G} [Eq. (24)] now includes a contribution arising from the use of $\bar{\gamma}(j)$ instead of $\hat{\gamma}(j)$. The result is

$$\delta\mathcal{G} = |\tilde{G} - \mathcal{G}| \leq |\tilde{G} - \bar{G}| + |\bar{G} - \bar{g}| + |\bar{g} - \mathcal{G}| \quad (72)$$

or

$$\delta\mathcal{G} \leq \delta G + \delta\bar{g} + \varepsilon_d, \quad (73)$$

where $\varepsilon_d \equiv |\bar{g} - \mathcal{G}|$ is the discretization error due to replacing \mathcal{G} by the Riemann sum $\mathcal{G}_R = \bar{g}$. Using Eqs. (67) and (71) gives

$$\delta\mathcal{G} = O(\varepsilon_1) + O(\varepsilon_\gamma) + O(\varepsilon_d). \quad (74)$$

As in the calculations following Eq. (52), a similar formula holds for the error $\delta\mathcal{I}$ in the estimate of the integral \mathcal{I} [Eq. (21)]:

$$\delta\mathcal{I} = O(\varepsilon_1) + O(\varepsilon_\gamma) + O(\varepsilon_d). \quad (75)$$

This error reduces to Eq. (58) when $\bar{\gamma}(j) \rightarrow \hat{\gamma}(j)$ as this causes the $O(\varepsilon_\gamma) \rightarrow 0$.

As seen above, $\varepsilon_1 > 2\pi/M_1$, $\varepsilon_d = O(1/M_2)$ [Eq. (51)], and $\varepsilon_\gamma > 1/M_3$, so that

$$\begin{aligned} M_1 &= O(1/\varepsilon_1), \\ M_2 &= O(1/\varepsilon_d), \\ M_3 &= O(1/\varepsilon_\gamma). \end{aligned} \quad (76)$$

Recalling that $m_1 = \log_2 M_1$, $m_2 = \lceil \log_2 M_2 \rceil$, and $m_3 = \log_2 M_3$, Eq. (76) gives

$$\begin{aligned} m_1 &= O(\log_2(1/\varepsilon_1)), \\ m_2 &= O(\log_2(1/\varepsilon_d)), \\ m_3 &= O(\log_2(1/\varepsilon_\gamma)). \end{aligned} \quad (77)$$

From Fig. 1 and Eq. (77) we see that the widths m_1 and $m_2 + 1$ of the two quantum registers appearing in the quantum circuit for the QPEA are determined by the errors ε_1 and ε_d , respectively. We shall see shortly that m_3 is also the width of a quantum register which, by Eq. (77), is determined by the error ε_γ . We will return to these formulas in Sec. III D.

Finally, we use the binary coefficients $\{\bar{\gamma}_i(j) | 0 \leq i \leq m_3 - 1\}$ with $0 \leq j \leq M_3 - 1$ to define the collection of CB states $|\bar{\gamma}(j)\rangle_{m_3}$:

$$|\bar{\gamma}(j)\rangle_{m_3} \equiv |\bar{\gamma}_0(j)\rangle \otimes \cdots \otimes |\bar{\gamma}_{m_3-1}(j)\rangle. \quad (78)$$

These states play an important role in the remainder of this section.

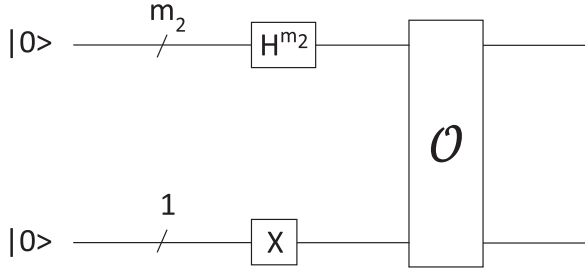


FIG. 2. Quantum circuit for A . The operator O is Novak's quantum oracle, H^{m_2} denotes transversal application of Hadamard gates to the qubits in the first register, and X is the Pauli X gate.

2. Quantum circuit

We have seen that for Novak's QIA, the operator A used to prepare the state $|\psi\rangle$ [Eqs. (40) and (41)] is

$$A = O(H^{m_2} \otimes X). \quad (79)$$

The circuit for A appears in Fig. 2. Implementing A thus reduces to implementing Novak's oracle O . The circuit for O appears in Fig. 3. The operator R is defined to have the action

$$\begin{aligned} R |\bar{\gamma}(j)\rangle_{m_3} \otimes |1\rangle_1 \\ &\equiv |\bar{\gamma}(j)\rangle_{m_3} \otimes U[\bar{\gamma}(j)]|1\rangle \\ &= |\bar{\gamma}(j)\rangle_{m_3} \otimes [\sqrt{G(\tau_j)}|1\rangle_1 + \sqrt{1-G(\tau_j)}|0\rangle_1]. \end{aligned} \quad (80)$$

Here $U[\bar{\gamma}(j)] \equiv \exp[i(\pi/2)\bar{\gamma}(j)Y]$, and Y is the Pauli Y operator. In going from the first to the second line, we have used Eq. (65) and

$$U[\bar{\gamma}(j)]|1\rangle_1 = \cos \frac{\pi}{2} \bar{\gamma}(j)|1\rangle_1 + \sin \frac{\pi}{2} \bar{\gamma}(j)|0\rangle_1. \quad (81)$$

The action for S is

$$S|j\rangle_{m_2} \otimes |0\rangle_{m_3} \equiv |j\rangle_{m_2} \otimes |\bar{\gamma}(j)\rangle_{m_3}, \quad (82)$$

where $|\bar{\gamma}(j)\rangle_{m_3}$ was defined in Eq. (78).

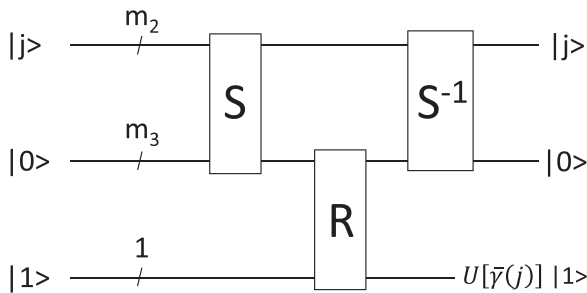


FIG. 3. Quantum circuit for O . The operators R and S are defined in the text. The second register is an ancilla register needed for intermediate operations. It is disentangled from the other registers by S^{-1} and discarded upon circuit completion. The operator $U[\bar{\gamma}(j)]$ is defined in Eq. (87).

We saw in Sec. II B that Novak's oracle O implemented the action:

$$\begin{aligned} O \left[\frac{1}{\sqrt{M_2}} \sum_{j=0}^{M_2-1} |j\rangle_{m_2} \otimes |1\rangle_1 \right] \\ &= \frac{1}{\sqrt{M_2}} \sum_{j=0}^{M_2-1} |j\rangle_{m_2} \\ &\quad \times \otimes [\sqrt{G(\tau_j)}|1\rangle_1 + \sqrt{1-G(\tau_j)}|0\rangle_1], \end{aligned} \quad (83)$$

where $g(\tau_j) \rightarrow G(\tau_j)$, since we must work with $\bar{\gamma}(j)$ instead of $\hat{\gamma}(j)$ (finite resources again). We now show that the circuit in Fig. 3 implements this action. Suppressing the normalization factor, the action of the circuit is

$$\begin{aligned} S^{-1}RS \left\{ \sum_{j=0}^{M_2-1} |j\rangle_{m_2} \otimes |0\rangle_{m_3} \otimes |1\rangle_1 \right\} \\ &= S^{-1}R \left\{ \sum_{j=0}^{M_2-1} |j\rangle_{m_2} \otimes |\bar{\gamma}(j)\rangle_{m_3} \otimes |1\rangle_1 \right\} \\ &= S^{-1} \left\{ \sum_{j=0}^{M_2-1} |j\rangle_{m_2} \otimes |\bar{\gamma}(j)\rangle_{m_3} \otimes U[\bar{\gamma}(j)]|1\rangle_1 \right\} \\ &= \sum_{j=0}^{M_2-1} |j\rangle_{m_2} \otimes |0\rangle_{m_3} \otimes U[\bar{\gamma}(j)]|1\rangle_1. \end{aligned} \quad (84)$$

Since the second register is not entangled with the other registers at the end of the circuit, it can be discarded. Reinstating the normalization factor, and suppressing the second register, we see that

$$\begin{aligned} S^{-1}RS \left\{ \sum_{j=0}^{M_2-1} |j\rangle_{m_2} \otimes |1\rangle_1 \right\} \\ &= \frac{1}{\sqrt{M_2}} \sum_{j=0}^{M_2-1} |j\rangle_{m_2} \\ &\quad \times \otimes [\sqrt{G(\tau_j)}|1\rangle_1 + \sqrt{1-G(\tau_j)}|0\rangle_1], \end{aligned} \quad (85)$$

where we have used Eq. (81). Comparing the RHSs of Eqs. (83) and (85) we see that the circuit in Fig. 3 does in fact implement O .

Finally, note that the circuit for A^{-1} applies the circuit for A in reverse order with $O \rightarrow O^{-1}$, and the circuit for O^{-1} is the same as that of O , but with $R \rightarrow R^{-1}$. We present the circuits for R and S , and their inverses, in Secs. III B and III C.

B. Quantum circuit for R

The circuit for R appears in Fig. 4. To see that it correctly implements R 's action,

$$R |\bar{\gamma}(j)\rangle_{m_3} \otimes |1\rangle_1 = |\bar{\gamma}\rangle_{m_3} \otimes U[\bar{\gamma}(j)]|1\rangle_1, \quad (86)$$

where

$$U[\bar{\gamma}(j)] \equiv \exp [i(\pi/2)\bar{\gamma}(j)Y] \quad (87)$$

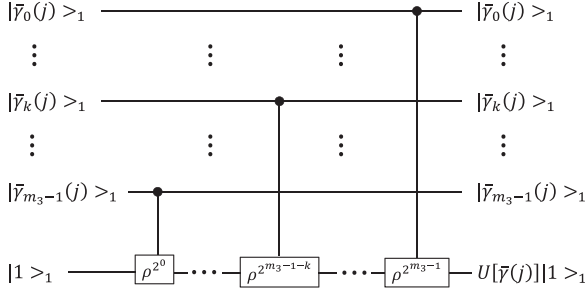


FIG. 4. Quantum circuit for R . The circuit acts on two registers. The first is an m_3 -qubit register and the second is a one-qubit register. The qubits in the first register act as control qubits, and the qubit in the second register is the target qubit. The operator ρ is defined in Eq. (90).

and

$$\bar{\gamma}(j) \equiv \bar{\gamma}_0(j) \left(\frac{1}{2}\right)^1 + \cdots + \bar{\gamma}_{m_3-1}(j) \left(\frac{1}{2}\right)^{m_3}, \quad (88)$$

we insert Eq. (88) into Eq. (87). This gives

$$\begin{aligned} U[\bar{\gamma}(j)] &= \otimes_{k=0}^{m_3-1} \exp \left[i \frac{\pi}{2^{k+2}} \bar{\gamma}_k(j) Y \right] \\ &= [\rho^{2^{m_3-1}}]^{\bar{\gamma}_0(j)} \cdots [\rho^{2^{m_3-1-k}}]^{\bar{\gamma}_k(j)} \cdots [\rho^{2^0}]^{\bar{\gamma}_{m_3-1}(j)}, \end{aligned} \quad (89)$$

where

$$\rho \equiv \exp \left[i \frac{\pi}{2^{m_3+1}} Y \right]. \quad (90)$$

We see that the k th factor in Eq. (89) corresponds to the controlled- $\rho^{2^{m_3-1-k}}$ gate in Fig. 4, with $0 \leq k \leq m_3 - 1$. Once the circuit has completed, $U[\bar{\gamma}(j)]$ has been applied to the initial state $|1\rangle_1$ of the second register, while the first register remains in the state $|\bar{\gamma}(j)\rangle_{m_3}$. This is the R action of Eq. (86). The circuit for R^{-1} applies the circuit for R in reverse order with $\rho \rightarrow \rho^{-1}$.

We see that the circuit for R applies up to $M_3 - 1$ controlled- ρ operations, and so

$$\#(\text{controlled-}\rho)_R = O(M_3), \quad (91)$$

and this is also the circuit depth

$$D(R) = O(M_3). \quad (92)$$

Finally, the circuit width is $W(R) = m_3 + 1$, and so

$$W(R) = O(\log_2 M_3). \quad (93)$$

C. Quantum circuit for S

The action for S appears in Eq. (82), where $|j\rangle_{m_2} = |j_0 \cdots j_{m_2-1}\rangle$ is a CB state, and $|\bar{\gamma}(j)\rangle_{m_3}$ is defined in Eq. (78). As discussed in Sec. III A 1, the angles $\bar{\gamma}(j)$ ($0 \leq j \leq M_2 - 1$) are known, as are the associated bit-values $\bar{\gamma}_0(j), \dots, \bar{\gamma}_{m_3-1}(j)$ [Eq. (62)]. In a slightly abbreviated notation, let $w(j)$ denote the weight of the bit-string

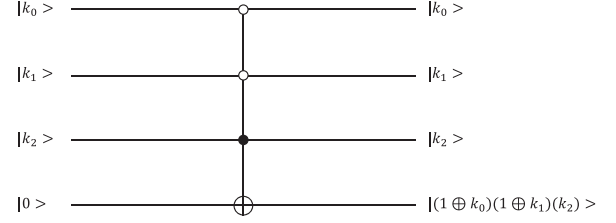


FIG. 5. Action of the multi-controlled NOT gate $C_j(X)$. For this circuit, $j = 4$ so that $j_0 = 0$, $j_1 = 0$, and $j_2 = 1$. Open (closed) circles indicate that the associated control qubit must be in the $|0\rangle$ ($|1\rangle$) state for the gate to act nontrivially, and \oplus denotes binary addition. Consequently, the open and closed circles ensure the circuit applies X to the target qubit only when $k_0 = 0$, $k_1 = 0$, and $k_2 = 1$ (viz. $|k\rangle = |j\rangle = |4\rangle$), and acts as the identity for all $|k\rangle \neq |4\rangle$.

$$\bar{\gamma}_0(j) \cdots \bar{\gamma}_{m_3-1}(j):$$

$$w(j) = \sum_{k=0}^{m_3-1} \bar{\gamma}_k(j). \quad (94)$$

It is thus the number of bit-values $\bar{\gamma}_k(j)$ that are equal to 1, and so satisfies the bound $w(j) \leq m_3$.

Note that to produce the action of Eq. (82), we simply need to flip $|0\rangle \rightarrow |1\rangle$ for the qubits in the second register whose qubit label k matches the bit labels k for which $\bar{\gamma}_k(j) = 1$. A key ingredient for doing this are the multi-controlled NOT gates $C_j(X)$, $0 \leq j \leq M_2 - 1$, which have m_2 control qubits and one target qubit. If the coefficients in the binary expansion of j are (j_0, \dots, j_{m_2-1}) , then the action of $C_j(X)$ on the CB states $|k\rangle_{m_2} \otimes |i\rangle_1$ is

$$\begin{aligned} C_j(X) |k\rangle_{m_2} \otimes |i\rangle_1 \\ = |k\rangle_{m_2} \otimes X^{(1 \oplus j_0 \oplus k_0) \times \cdots \times (1 \oplus j_{m_2-1} \oplus k_{m_2-1})} |i\rangle_1, \end{aligned} \quad (95)$$

where \oplus denotes binary addition. Thus $C_j(X)$ applies X when $|k\rangle_{m_2} = |j\rangle_{m_2}$, and acts as the identity when $|k\rangle_{m_2} \neq |j\rangle_{m_2}$. For example, suppose $m_2 = 3$ and $j = 4$ so that $j_0 = 0$, $j_1 = 0$, and $j_2 = 1$. Figure 5 contains the circuit that applies $C_4(X)$. Open (closed) circles in Fig. 5 indicate that the associated control qubits must be in the $|0\rangle_1$ ($|1\rangle_1$) states or the $C_4(X)$ gate will not act. It is clear that the final state is $|k\rangle_3 \otimes |1\rangle_1$ when $k = j$, and is $|k\rangle_3 \otimes |0\rangle_1$ when $k \neq j$. This is the action of Eq. (95).

The next step towards implementing the action of Eq. (82) is applying $C_j(X)$ $w(j)$ times, where the control register contains m_2 qubits and the target register contains m_3 qubits. The $w(j)$ target qubits for the $C_j(X)$ gates have qubit labels k that match the bit labels k for which $\bar{\gamma}_k(j) = 1$. An example will help to illustrate this. Suppose that $m_2 = 3$, $m_3 = 2$, $j = 4$, and $|\bar{\gamma}(4)\rangle_2 = |11\rangle$. Thus $|4\rangle_3 = |001\rangle$, the weight of $|\bar{\gamma}(4)\rangle_2$ is $w(4) = 2$, and $\bar{\gamma}_k(4) = 1$ for $k = 0$ and 1. The circuit in Fig. 6 thus applies two $C_4(X)$ gates whose target qubits are qubit 0 and qubit 1 in the second register. We see that when $k_0 = 0$, $k_1 = 0$, and $k_2 = 1$, the final state of the two registers is $|001\rangle \otimes |11\rangle = |4\rangle_3 \otimes |\bar{\gamma}(4)\rangle$ as desired. For all other values of k_0 , k_1 , and k_2 , the final state is $|k_0 k_1 k_2\rangle \otimes |00\rangle = |k\rangle_3 \otimes |00\rangle$ so that the circuit acts as the identity, as desired.

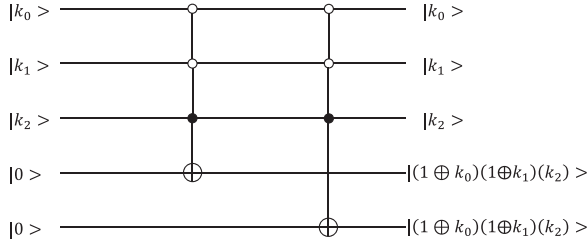


FIG. 6. Building block circuit for implementing S —illustrative example. For this circuit, $j \equiv 4$, and so $j_0 = 0$, $j_1 = 0$, $j_2 = 1$. We also set $|\bar{\gamma}(4)\rangle = |11\rangle$ so that $w(4) = 2$. The circuit thus applies two $C_4(X)$ gates, and the target qubits are qubit 0 and qubit 1 in the second register. The circuit leaves the second register in the state $|\bar{\gamma}(4)\rangle_2$ when the input state $|k_0 k_1 k_2\rangle = |001\rangle = |j\rangle_3 = |4\rangle_3$, and acts as the identity when $|k\rangle_3 \neq |4\rangle_3$.

Since the circuit in Fig. 6 applies a linear operation, its action on a superposition of states $|k\rangle_3 \otimes |00\rangle$ follows from linearity.

To construct a circuit that implements S , we use as the basic building block the strategy underlying the construction of the circuit in Fig. 6. For each j value ($0 \leq j \leq M_2 - 1$) we apply $C_j(X)$ gates $w(j)$ times as just described. As we have seen, the gates for a specific j value only act nontrivially when the first register is in the CB state $|j\rangle_{m_2} = |j_0 \cdots j_{m_2-1}\rangle$. The circuit sequentially applies $C_j(X)$ gates $w(j)$ times for all j values to produce the S -action of Eq. (82). To illustrate the S -circuit construction, suppose again $m_2 = 3$, $m_3 = 2$, and

$$\begin{aligned} |\bar{\gamma}(0)\rangle &= |01\rangle \rightarrow w(0) = 1, \\ |\bar{\gamma}(1)\rangle &= |10\rangle \rightarrow w(1) = 1, \\ |\bar{\gamma}(2)\rangle &= |00\rangle \rightarrow w(2) = 0, \\ |\bar{\gamma}(3)\rangle &= |11\rangle \rightarrow w(3) = 2, \\ |\bar{\gamma}(4)\rangle &= |01\rangle \rightarrow w(4) = 1, \\ |\bar{\gamma}(5)\rangle &= |10\rangle \rightarrow w(5) = 1, \\ |\bar{\gamma}(6)\rangle &= |11\rangle \rightarrow w(6) = 2, \\ |\bar{\gamma}(7)\rangle &= |00\rangle \rightarrow w(7) = 0. \end{aligned} \quad (96)$$

The S -circuit for this example appears in Fig. 7. When the input to the first register is $|k\rangle = |k_0 k_1 k_2\rangle$, then all $C_j(X)$ gates

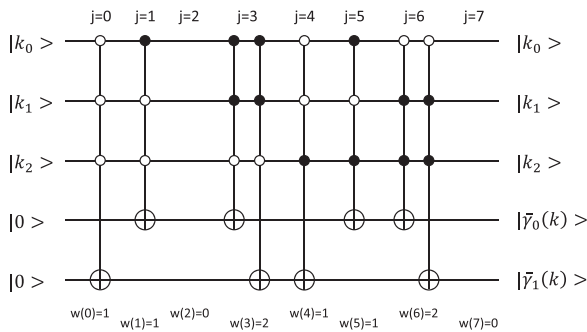


FIG. 7. Quantum circuit for S —illustrative example. For this circuit, $m_2 = 3$, $m_3 = 2$, and the $\bar{\gamma}(j)$ ($0 \leq j \leq 7$) are given in Eq. (96). The circuit sequentially applies $C_j(X)$ gates $w(j)$ times, starting with $j = 0$ and ending with $j = 7$. The circuit takes $|k\rangle_3 \otimes |00\rangle$ to $|k\rangle_3 \otimes |\bar{\gamma}(k)\rangle_2$.

act as the identity except for the $w(j)$ gates that have $j = k$, and so they are triggered by the input values k_0 , k_1 , and k_2 . These gates toggle the second register initial state so as to produce $|\bar{\gamma}(k)\rangle$ as the second register final state. By linearity, the circuit produces the action

$$\frac{1}{\sqrt{C}} \sum_{k=0}^{M_2-1} |k\rangle_{m_2} \otimes |0\rangle_{m_3} \rightarrow \frac{1}{\sqrt{C}} \sum_{k=0}^{M_2-1} |k\rangle_{m_2} \otimes |\bar{\gamma}(k)\rangle_{m_3}, \quad (97)$$

which is the desired S -action. Finally, since all gates in the circuit for S are their own inverse, the circuit for S^{-1} is obtained by reversing the sequence of gates in the circuit for S .

The number η of multi-controlled NOT gates is

$$\eta = \sum_{k=0}^{M_2-1} w(k) \leq M_2 \log_2 M_3 = O(M_2 \log_2 M_3), \quad (98)$$

where we have used that $w(k) \leq m_3 = \log_2(M_3)$. In the Appendix, we show that each multi-controlled NOT gate can be implemented using $2m_2 - 3 = 2 \log_2 M_2 - 3$ Toffoli gates and $\log_2 M_2 - 2$ ancilla qubits when $m_2 \geq 2$. We do not count the X gates needed to implement the open circle triggering as these are Clifford gates, which are easier to implement than the non-Clifford Toffoli gates. The number of Toffoli gates $\mathcal{T}(S)$ is then

$$\mathcal{T}(S) = (2 \log_2 M_2 - 3)\eta = O(M_2 \log_2 M_2 \log_2 M_3). \quad (99)$$

This is also the circuit depth $D(S) = \mathcal{T}(S)$. The circuit width $W(S)$, including the Toffoli ancilla, is

$$\begin{aligned} W(S) &= \log_2 M_2 + \log_2 M_3 + (\log_2 M_2 - 2) \\ &= O(\log_2 M_2) + O(\log_2 M_3). \end{aligned} \quad (100)$$

Having shown how to implement the circuits for R , S , and S^{-1} , the circuits appearing in Figs. 2 and 3 then show how to implement the state preparation operator A and the oracle O , respectively.

D. Resource requirements

Here we gather together the resource requirements found for R , S , and S^{-1} in Secs. III B and III C, and we use them to determine the resources needed to implement Novak's oracle O and the state preparation operator A . We make use of Eqs. (76) and (77) to rewrite these requirements in terms of the error tolerances ε_1 , ε_d , and ε_γ . We do this to underscore that it is these tolerances that drive the resource demands for O and A . Recall that (i) ε_1 [Eq. (45)] is the error tolerance for the error in the QAEA estimate for the Riemann sum \mathcal{G}_R ; (ii) ε_d [Eq. (50)] is the error arising from replacing the integral \mathcal{I} by the Riemann sum \mathcal{I}_R ; and (iii) ε_γ is the error tolerance [Eq. (71)] for the truncation error $\delta\hat{\gamma}(j)$.

Resources for R: From Sec. III B, the depth of the circuit for R goes as

$$D(R) = O(M_3) = O(1/\varepsilon_\gamma); \quad (101)$$

its width goes as

$$W(R) = O(\log_2 M_3) = O(\log_2 1/\varepsilon_\gamma); \quad (102)$$

and the number of controlled small-angle rotations goes as

$$\#(\text{controlled-}\rho)_R = O(M_3) = O(1/\varepsilon_\gamma). \quad (103)$$

Resources for S and S^{-1} : From Sec. III C, the depth of the circuits for S and S^{-1} goes as

$$D(S) = D(S^{-1}) = O((1/\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)); \quad (104)$$

its width goes as

$$W(S) = W(S^{-1}) = O(\log_2 1/\varepsilon_d) + O(\log_2 1/\varepsilon_\gamma); \quad (105)$$

and the number of Toffoli gates goes as

$$\mathcal{T}(S) = \mathcal{T}(S^{-1}) = O((1/\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)). \quad (106)$$

Resources for O : From Fig. 3, the depth of the circuit for O goes as

$$\begin{aligned} D(O) &= D(S) + D(S^{-1}) + D(R) \\ &= O((1/\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)) + O(1/\varepsilon_\gamma); \end{aligned} \quad (107)$$

its width goes as

$$\begin{aligned} W(O) &= W(S) + W(R) \\ &= O(\log_2 1/\varepsilon_d) + O(\log_2 1/\varepsilon_\gamma); \end{aligned} \quad (108)$$

the number of Toffoli gates used goes as

$$\begin{aligned} \mathcal{T}(O) &= \mathcal{T}(S) + \mathcal{T}(S^{-1}) \\ &= O((1/\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)); \end{aligned} \quad (109)$$

and the number of controlled small-angle rotations goes as

$$\#(\text{controlled-}\rho)_O = \#(\text{controlled-}\rho)_R = O(1/\varepsilon_\gamma). \quad (110)$$

Resources for A : From Fig. 2 the depth of the circuit for A goes as

$$\begin{aligned} D(A) &= D(O) \\ &= O((1/\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)) + O(1/\varepsilon_\gamma); \end{aligned} \quad (111)$$

its width goes as

$$W(A) = W(O) = O(\log_2 1/\varepsilon_d) + O(\log_2 1/\varepsilon_\gamma); \quad (112)$$

the number of Toffoli gates used goes as

$$\mathcal{T}(A) = \mathcal{T}(O) = O((1/\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)), \quad (113)$$

and the number of controlled small-angle rotations goes as

$$\#(\text{controlled-}\rho)_A = \#(\text{controlled-}\rho)_O = O(1/\varepsilon_\gamma). \quad (114)$$

IV. IMPLEMENTING Q

In this section, we present the circuit implementation of the Grover-like oracle Q . The circuit is presented in Sec. IV A and makes use of operators U_0 and U_{n_0} , whose circuit implementations appear in Secs. IV B and IV C, respectively. Finally, Sec. IV D gathers together the resource requirements needed to implement Q .

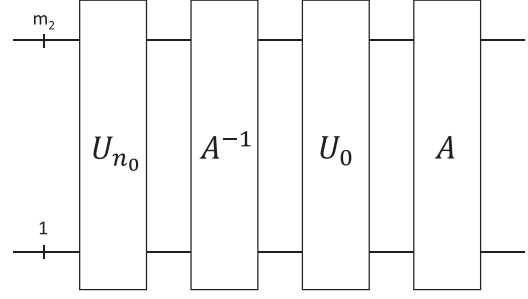


FIG. 8. Quantum circuit for oracle Q . The circuit first applies the inversion operator U_{n_0} , then applies the operator A^{-1} which maps the state $|\psi\rangle = \sqrt{a}|n_1\rangle + \sqrt{1-a}|n_0\rangle$ to the $m_2 + 1$ qubit CBS $|0\rangle$. The inversion operator U_0 is then applied, and finally, the operator A is applied. (i) The circuits for U_0 and U_{n_0} appear in Secs. IV B and IV C, respectively; while (ii) the circuits for A and A^{-1} appear in Sec. III A 2.

A. Quantum circuit for Q

The action of the oracle Q on the 2D subspace \mathcal{H}_ψ spanned by $|n_0\rangle$ and $|n_1\rangle$ is [Eqs. (36) and (37)]

$$Q = A U_0 A^{-1} U_{n_0}, \quad (115)$$

where

$$U_0 = I - 2|0\rangle\langle 0|, \quad (116a)$$

$$U_{n_0} = I - 2|n_0\rangle\langle n_0|, \quad (116b)$$

and [Eq. (41)]

$$A = O[H^{m_2} \otimes X]. \quad (117)$$

The $m_2 + 1$ qubit state $|n_0\rangle$ appears in Eq. (30), and $|0\rangle = |0 \cdots 0\rangle_{m_2} \otimes |0\rangle_1$ is an $m_2 + 1$ qubit CBS.

The action of U_0 on the $m_2 + 1$ qubit CBS $|j\rangle = |j_0 \cdots j_{m_2-1}\rangle \otimes |j_{m_2}\rangle$ is easily seen to be

$$U_0|j\rangle = \begin{cases} -|0\rangle & (j = 0), \\ |j\rangle & (j \neq 0). \end{cases} \quad (118)$$

Similarly, the action of U_{n_0} on the states $|n_0\rangle$ and $|n_1\rangle$ is

$$U_{n_0}|n_0\rangle = -|n_0\rangle, \quad (119a)$$

$$U_{n_0}|n_1\rangle = |n_1\rangle. \quad (119b)$$

The circuit for Q follows from Eq. (115) and appears in Fig. 8, while the circuits for A and A^{-1} appear in Sec. III A 2. We will determine the resource requirements for the circuit implementation of Q in Sec. IV D, after presenting the circuits for U_0 and U_{n_0} .

B. Quantum circuit for U_0

The circuit that implements U_0 appears in Fig. 9. To see that it implements the action of Eq. (118), we step through the circuit's operation. The circuit acts on two registers. The first register contains $m_2 + 1$ qubits whose CBSs are $|j\rangle = |j_0 \cdots j_{m_2}\rangle$, where $0 \leq j \leq 2M_2 - 1$ and $M_2 = 2^{m_2}$. The second register contains a single ancilla qubit.

Let the initial state $|\psi_0\rangle$ of the circuit have the first register in an arbitrary superposition of CBSs, and the second register

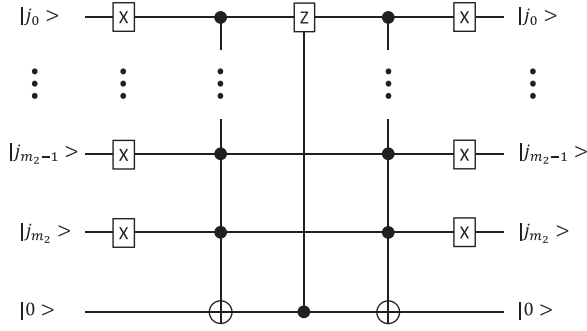


FIG. 9. Quantum circuit for U_0 . The circuit begins by applying Pauli X gates transversally to the $m_2 + 1$ qubits in the first register whose CBS are $|j\rangle = |j_0 \cdots j_{m_2}\rangle$. The circuit then applies the multi-controlled NOT gate $C_{2M_2-1}(X)$ that applies a Pauli X gate to the single qubit in the second register when all control qubits are in the $|1\rangle_1$ state, and it acts as the identity for all other control inputs. A controlled- Z gate is applied using the ancilla qubit in the second register as the control qubit and the 0-qubit in the first register as the target qubit. To disentangle the second register from the first, the multi-controlled NOT gate $C_{2M_2-1}(X)$ is applied again. The second register is then discarded. Finally, Pauli X gates are transversally applied to the first register.

in the $|0\rangle$ state:

$$|\psi_0\rangle = \sum_{j=0}^{2M_2-1} a(j) |j_0 \cdots j_{m_2}\rangle \otimes |0\rangle. \quad (120)$$

Applying the Pauli X gates transversally to the first register gives the state

$$|\psi_1\rangle = \sum_{j=0}^{2M_2-1} a(j) |(1 \oplus j_0) \cdots (1 \oplus j_{m_2})\rangle \otimes |0\rangle. \quad (121)$$

The multi-controlled NOT gates in Fig. 9 are $C_{2M_2-1}(X)$, which only act when all control qubits are in the $|1\rangle_1$ state (viz. all closed circles). The action of the first multi-controlled NOT gate gives the state

$$|\psi_2\rangle = \sum_{j=0}^{2M_2-1} a(j) |(1 \oplus j_0) \cdots (1 \oplus j_{m_2})\rangle \otimes |\chi(j)\rangle, \quad (122)$$

where

$$\begin{aligned} \chi(j) &\equiv (1 \oplus j_0) \times \cdots \times (1 \oplus j_c) \\ &= \begin{cases} 1 & (j_0 = \cdots = j_{m_2} = 0), \\ 0 & (\text{otherwise}). \end{cases} \end{aligned} \quad (123)$$

The circuit then applies a controlled- Z gate to the ancilla qubit and the 0-qubit in the first register. Recall that the action of the controlled- Z gate, Z^λ ($\lambda = 0, 1$), is to apply the Pauli Z gate when the control qubit is in the $|\lambda = 1\rangle_1$ state, and the identity when in the $|\lambda = 0\rangle_1$ state. The action of Z^λ on the state $|\psi_2\rangle$ is then

$$\begin{aligned} |\psi_3\rangle &= \sum_{j=0}^{2M_2-1} (-1)^{\chi(j)(1 \oplus j_0)} \\ &\times a(j) |(1 \oplus j_0) \cdots (1 \oplus j_{m_2})\rangle \otimes |\chi(j)\rangle. \end{aligned} \quad (124)$$

To disentangle the second register from the first, the circuit applies the second multi-controlled NOT gate, $C_{2M_2-1}(X)$, to $|\psi_3\rangle$. The result is

$$|\psi_4\rangle = \sum_{j=0}^{2M_2-1} (-1)^{\chi(j)(1 \oplus j_0)} a(j) |(1 \oplus j_0) \cdots (1 \oplus j_{m_2})\rangle \otimes |0\rangle. \quad (125)$$

The second register can now be safely discarded. Applying the final Pauli X gates to the first register gives

$$|\psi_f\rangle = \sum_{j=0}^{2M_2-1} (-1)^{\chi(j)(1 \oplus j_0)} a(j) |j_0 \cdots j_{m_2}\rangle. \quad (126)$$

To see that this is the action of Eq. (118), suppose that $a(0) = 1$ and $a(j) = 0$ for all $j \neq 0$ so that the initial state is $|\psi_0\rangle = |0\rangle$. From Eq. (126), the final state is then

$$|\psi_f\rangle = -|0\rangle, \quad (127)$$

and the circuit thus maps $|0\rangle \rightarrow -|0\rangle$ as in Eq. (118). Similarly, let $a(j) = 1$ for $j \neq 0$ and $a(k) = 0$ for all $k \neq j$ so that $|\psi_0\rangle = |j\rangle$. Then, from Eq. (126),

$$|\psi_f\rangle = |j\rangle, \quad (128)$$

and the circuit thus maps $|j\rangle \rightarrow |j\rangle$ as in Eq. (118). Thus the circuit in Fig. 9 has reproduced the action of Eq. (118).

The resource requirements needed to implement U_0 follow from Fig. 9. Since the Clifford gates add three steps to the circuit depth in Fig. 9, independent of the number of qubits $m_2 + 1$ in the first register, they will have a negligible impact on the scaling of the circuit depth. Instead, the depth is controlled by the two multi-controlled NOT gates. From the Appendix we know that the depth of the circuit for a multi-controlled NOT gate is $O(\log_2 M_2) = O(\log_2(1/\varepsilon_d))$. Thus the U_0 circuit depth is

$$D(U_0) = O(\log_2(1/\varepsilon_d)). \quad (129)$$

Similarly, the U_0 circuit width is controlled by the width of the multi-controlled NOT gate circuit. From the Appendix, this is $W(C_{2M_2-1}(X)) = O(\log_2(1/\varepsilon_d))$. Thus

$$W(U_0) = O(\log_2(1/\varepsilon_d)). \quad (130)$$

The number of Toffoli gates needed to implement U_0 is equal to the number used to implement the multi-controlled NOT gates. From the Appendix we know that $\mathcal{T}(C_{2M_2-1}(X)) = O(\log_2(1/\varepsilon_d))$ so that

$$\mathcal{T}(U_0) = O(\log_2(1/\varepsilon_d)). \quad (131)$$

Finally, no controlled- ρ rotations are used to implement U_0 so that

$$\#(\text{controlled-}\rho)_{U_0} = 0. \quad (132)$$

C. Quantum circuit for U_{n_0}

The action U_{n_0} that applies to the states $|n_0\rangle$ and $|n_1\rangle$ (that span the 2D subspace \mathcal{H}_ψ) is given in Eq. (119). The quantum circuit that implements this action appears in Fig. 10. The circuit acts on two registers: the first contains m_2 qubits, while the second contains one qubit.

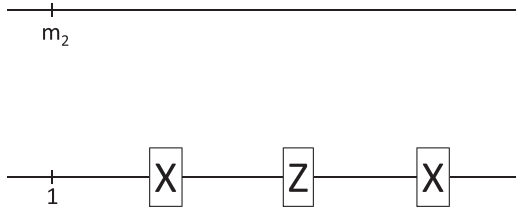


FIG. 10. Quantum circuit for U_{n_0} . The quantum circuit acts on two registers: the first contains m_2 qubits and the second contains a single qubit. The circuit applies Pauli X and Z gates to the qubit in the second register as shown. The qubits in the first register are not acted upon.

From Sec. II B,

$$\begin{aligned} |n_1\rangle &= \frac{1}{\sqrt{M_2}} \sum_{j=0}^{M_2-1} \sqrt{\frac{g(\tau_j)}{\bar{g}}} |j\rangle \otimes |1\rangle \\ &\equiv |\gamma\rangle \otimes |1\rangle, \end{aligned} \quad (133a)$$

$$\begin{aligned} |n_0\rangle &= \frac{1}{\sqrt{M_2}} \sum_{j=0}^{M_2-1} \sqrt{\frac{1-g(\tau_j)}{1-\bar{g}}} |j\rangle \otimes |0\rangle \\ &\equiv |\beta\rangle \otimes |0\rangle. \end{aligned} \quad (133b)$$

The circuit in Fig. 10 applies the operator XZX to the qubit in the second register so that

$$\begin{aligned} XZX|n_1\rangle &= |\gamma\rangle \otimes XZX|1\rangle \\ &= |\gamma\rangle \otimes |1\rangle \\ &= |n_1\rangle \end{aligned} \quad (134)$$

and

$$\begin{aligned} XZX|n_0\rangle &= |\beta\rangle \otimes XZX|0\rangle \\ &= -|\beta\rangle \otimes |0\rangle \\ &= -|n_0\rangle. \end{aligned} \quad (135)$$

Equations (134) and (135) are the action appearing in Eq. (119), and so the circuit in Fig. 10 does, in fact, implement U_{n_0} .

The resource requirements for U_{n_0} are easily determined from Fig. 10. The circuit depth is 3, and so

$$D(U_{n_0}) = O(1). \quad (136)$$

Its width is $m_2 + 1$, and so

$$W(U_{n_0}) = O(m_2) = O(\log_2 1/\varepsilon_d). \quad (137)$$

Finally, the number of Toffoli gates and controlled- ρ gates is

$$\mathcal{T}(U_{n_0}) = \#(\text{controlled-}\rho)_{U_{n_0}} = 0. \quad (138)$$

D. Resource requirements

The resources needed to implement Q follow from Fig. 8. The circuit depth $D(Q)$, width $W(Q)$, number of Toffoli gates $\mathcal{T}(Q)$, and number of controlled- ρ gates needed are

$$D(Q) = D(U_{n_0}) + D(A^{-1}) + D(U_0) + D(A), \quad (139a)$$

$$W(Q) = W(U_{n_0}) + W(A^{-1}) + W(U_0) + W(A), \quad (139b)$$

$$\mathcal{T}(Q) = \mathcal{T}(U_{n_0}) + \mathcal{T}(A^{-1}) + \mathcal{T}(U_0) + \mathcal{T}(A) \quad (139c)$$

$$\#(\text{controlled-}\rho)_Q$$

$$\begin{aligned} &= \#(\text{controlled-}\rho)_{U_{n_0}} + \#(\text{controlled-}\rho)_{A^{-1}} \\ &\quad + \#(\text{controlled-}\rho)_{U_0} + \#(\text{controlled-}\rho)_A. \end{aligned} \quad (139d)$$

Sections III D, IV B, and IV C give the depth, width, number of Toffoli gates, and controlled- ρ gates for A , U_0 , and U_{n_0} , respectively. The resources needed for A^{-1} are clearly the same as for A .

For the circuit depth, we see that the depths of A and A^{-1} dominate the depths of U_0 and U_{n_0} . Thus,

$$D(Q) = O[(1/\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)] + O(1/\varepsilon_\gamma). \quad (140)$$

The widths $W(A)$, $W(A^{-1})$, $W(U_0)$, and $W(U_{n_0})$ are comparable, and so

$$W(Q) = O(\log_2 1/\varepsilon_d) + O(\log_2 1/\varepsilon_\gamma). \quad (141)$$

The number of Toffoli gates $\mathcal{T}(A)$ and $\mathcal{T}(A^{-1})$ dominate $\mathcal{T}(U_0)$ and $\mathcal{T}(U_{n_0})$, and so

$$\mathcal{T}(Q) = O[(1/\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)]. \quad (142)$$

Finally, only A and A^{-1} make use of controlled- ρ gates, and so

$$\#(\text{controlled-}\rho)_Q = O(1/\varepsilon_\gamma). \quad (143)$$

V. IMPLEMENTING $\Lambda(Q)$

In this section, we present the circuit implementation for our final oracle $\Lambda(Q)$. The circuit for $\Lambda(Q)$ is presented in Sec. V A and makes use of the controlled- Q operator. The circuit for this latter operator appears in Sec. V B and makes use of controlled- U_{n_0} and controlled- U_0 operators whose circuit implementations appear in Secs. V C and V D, respectively. Finally, Sec. V E gathers together the resource requirements needed to implement $\Lambda(Q)$.

A. Quantum circuit for $\Lambda(Q)$

The action of $\Lambda(Q)$ on CBS is

$$\Lambda(Q)|j\rangle_{m_1} \otimes |k\rangle_{m_2+1} = |j\rangle_{m_1} \otimes Q^j|k\rangle_{m_2+1}, \quad (144)$$

where $0 \leq j \leq M_1 - 1$, $M_1 = 2^{m_1}$, $0 \leq k \leq 2M_2 - 1$, and $M_2 = 2^{m_2}$. We begin with the binary decomposition of j ,

$$j = j_0 2^0 + \dots + j_{m_1-1} 2^{m_1-1}, \quad (145)$$

where the coefficients j_i are binary variables that take values of 0 or 1. We can thus write

$$Q^j = \prod_{i=0}^{m_1-1} (Q^{2^i})^{j_i}. \quad (146)$$

Note that each factor on the RHS of Eq. (146) is a controlled operation. For example, the i th factor applies Q^{2^i} when $j_i = 1$, and applies the identity when $j_i = 0$. Thus the binary coefficients $\{j_i\}$ control whether the operators $\{Q^{2^i}\}$ are applied. The quantum circuit in Fig. 11 makes use of this observation to apply the RHS of Eq. (146) and thus applies the $\Lambda(Q)$ -action

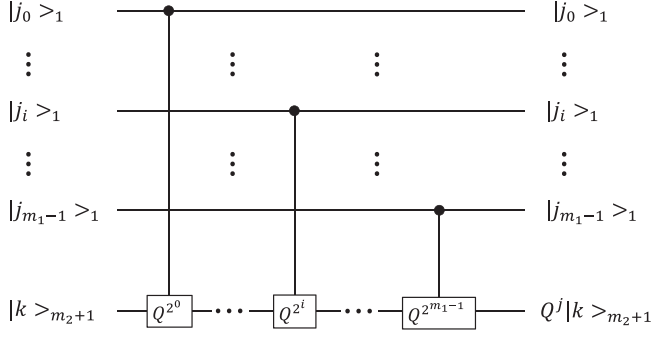


FIG. 11. Quantum circuit for $\Lambda(Q)$. The quantum circuit acts on two registers: the first contains m_1 qubits and the second contains $m_2 + 1$ qubits. The circuit applies up to $M_1 - 1$ controlled- Q operations, where $M_1 = 2^{m_1}$. We show how to implement the controlled- Q operation in Sec. VB.

given in Eq. (144). Specifically, each controlled operation in Fig. 11 applies one of the factors in Eq. (146), with the i th qubit in the first register controlling whether the operator Q^{2^i} is applied. At the end of the circuit, all factors in Eq. (146) have been applied and so the circuit applies $\Lambda(Q)$. The circuit acts on two registers: the first contains m_1 qubits, while the second contains $m_2 + 1$ qubits. The initial state for the circuit in Fig. 11 assumes the first and second registers are in the CBS $|j\rangle_{m_1} = |j_0 \cdots j_{m_1-1}\rangle_{m_1}$ and $|k\rangle_{m_2+1} = |k_0 \cdots k_{m_2}\rangle_{m_2+1}$, respectively. Because the circuit applies a linear operation, its action on an arbitrary superposition of CBS follows from linearity. As seen in Fig. 11, the $\Lambda(Q)$ circuit applies up to $M_1 - 1$ controlled- Q operations. We present the circuit implementation of the controlled- Q operation in Sec. VB.

B. Quantum circuit for controlled- Q

From Sec. IIC 1, $Q = AU_0A^{-1}U_{n_0}$. We now show that the circuit in Fig. 12 implements a controlled- Q operation (Q^λ). The circuit acts on two registers. The first contains a single qubit which acts as the control qubit, while the second contains $m_2 + 1$ qubits and is the target of the controlled- Q operation. Note that when the control qubit is in the $|\lambda = 0\rangle$ state, the controlled- U_{n_0} and controlled- U_0 operations act as the identity. The circuit then applies the identity operation

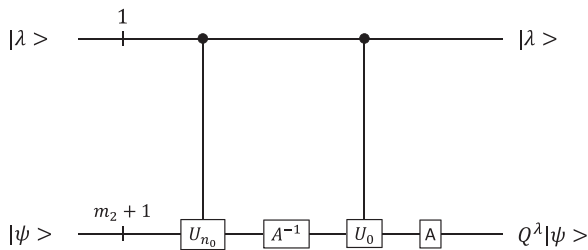


FIG. 12. Quantum circuit for controlled- Q . The quantum circuit acts on two registers: the first contains a single qubit which acts as the control qubit, and the second contains $m_2 + 1$ qubits which are the target of the controlled- Q operation. The circuit applies (i) controlled- U_{n_0} and controlled- U_0 operations whose circuit implementations appear in Secs. VC and VD, respectively; and (ii) A and A^{-1} operations whose circuit implementations appear in Sec. III A 2.

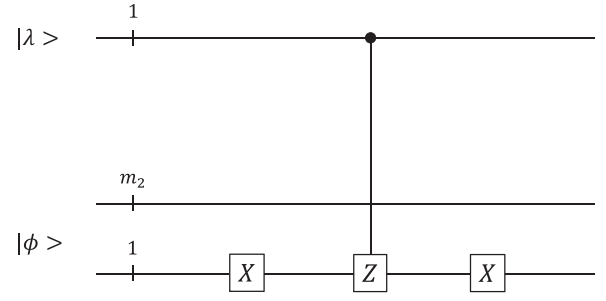


FIG. 13. Quantum circuit for controlled- U_{n_0} . The quantum circuit acts on two registers. The first contains a single qubit which acts as the control qubit, and the second is composed of two subregisters: subregister 2A contains m_2 qubits, and subregister 2B contains a single qubit. The composite second register is the target of the controlled- U_{n_0} operation. The circuit applies a Pauli X gate to the qubit in subregister 2B. It then applies a controlled- Z operation to the qubit in the first register and the qubit in subregister 2B. Finally, an X gate is applied to the qubit in subregister 2B.

$AA^{-1} = I$ to the second register, which is the desired operation when the control qubit is in the $|\lambda = 0\rangle$ state. When $|\lambda = 1\rangle$, the controlled- U_{n_0} and controlled- U_0 operations are applied and the total operation applied to the second register is $AU_0A^{-1}U_{n_0} = Q$ as desired. We see that the circuit in Fig. 12 does indeed apply a controlled- Q operation. Sections VC and VD present the quantum circuits that implement the controlled- U_{n_0} and controlled- U_0 operations, respectively. The circuit implementations for A and A^{-1} appear in Sec. III A 2.

C. Quantum circuit for controlled- U_{n_0}

We saw in Sec. IVC that U_{n_0} acts on two registers, the first containing m_2 qubits, the second containing a single qubit, and that it applied the operator XZX to the qubit in the second register. With this in mind, we now show that the circuit in Fig. 13 implements a controlled- U_{n_0} operation ($U_{n_0}^\lambda$). The circuit acts on two registers. The first contains a single qubit that acts as the control qubit of $U_{n_0}^\lambda$, and the second contains $m_2 + 1$ qubits and is the target. The second register is composed of two subregisters: the first (register 2A) contains m_2 qubits, and the second (register 2B) contains a single qubit.

From Fig. 13 we see that the circuit first applies a Pauli X gate to the qubit in subregister 2B. It then applies a controlled- Z gate between the qubit in register 1 and the qubit in subregister 2B. Finally, an X gate is applied to the qubit in subregister 2B. When the control qubit is in the state $|\lambda = 0\rangle$, the controlled- Z gate does not act and so the identity operation $X^2 = I$ is applied to the composite second register. This is the desired action when the control qubit is in the state $|\lambda = 0\rangle$. When the control qubit is in the state $|\lambda = 1\rangle$, the controlled- Z gate applies a Pauli Z gate to the qubit in subregister 2B. The total operation applied to subregister 2B is $XZX = U_{n_0}$, which is the desired action when the control qubit is in the state $|\lambda = 1\rangle$. The circuit thus applies a controlled- U_{n_0} operation.

The resource requirements follow easily from Fig. 13. The circuit depth is

$$D(U_{n_0}^\lambda) = 3 = O(1). \quad (147)$$

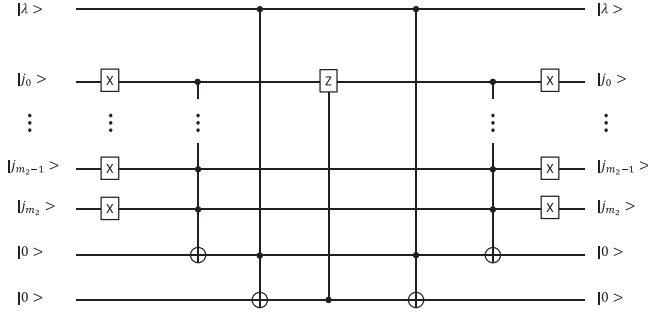


FIG. 14. Quantum circuit for controlled- U_0 (U_0^λ). The circuit acts on two registers and two ancilla qubits. The ancilla qubits are discarded upon completion of the circuit. The first register contains a single qubit and the second contains $m_2 + 1$ qubits. Comparing the circuit here with the circuit for U_0 in Fig. 9, we see that they are nearly identical. The circuit here has added a one-qubit first register that acts as the control for the U_0^λ operation, a second ancilla qubit, and two Toffoli gates that use the qubit in the first register and the first ancilla qubit as controls, and the second ancilla qubit as the target.

The circuit width is

$$W(U_{n_0}^\lambda) = m_2 + 2 = O(\log_2 1/\varepsilon_d). \quad (148)$$

Finally, the number of Toffoli gates and controlled- ρ gates is zero:

$$\mathcal{T}(U_{n_0}^\lambda) = 0, \quad (149)$$

$$\#(\text{controlled-}\rho)_{U_{n_0}^\lambda} = 0. \quad (150)$$

D. Quantum circuit for controlled- U_0

The U_0 -action on an $m_2 + 1$ qubit CBS is given by Eq. (118). With this in mind, we now show that the circuit in Fig. 14 implements a controlled- U_0 operation (U_0^λ). The circuit acts on two registers. The first is a one-qubit register that acts as the control qubit for U_0^λ , while the second register contains $m_2 + 1$ qubits and is the target. It also contains two ancilla qubits which are discarded at the end of the circuit. To see that the circuit applies U_0^λ , we step through its execution.

To begin, we assume that the circuit's initial state $|\psi_0\rangle$ has the first and second registers in an arbitrary state, while the two ancilla qubits are both in the state $|0\rangle$:

$$|\psi_0\rangle = \sum_{\lambda=0}^1 \sum_{j=0}^{2M_2-1} a(j, \lambda) |\lambda\rangle |j_0 \cdots j_{m_2}\rangle |0\rangle |0\rangle. \quad (151)$$

We have suppressed the tensor product symbol, and we will suppress the summation symbols for the duration of the calculation. At the end of the calculation, we will choose values for the coefficients $a(j, \lambda)$ to show that U_0^λ has been applied.

The circuit begins by applying transversal X gates to the target register. The resulting state is

$$|\psi_1\rangle = a(j, \lambda) |\lambda\rangle |(1 \oplus j_0) \cdots (1 \oplus j_{m_2})\rangle |0\rangle |0\rangle. \quad (152)$$

The circuit then applies the multi-controlled NOT gate $C_{2M_2-1}(X)$ using the second register as the control and the first ancilla qubit as the target. This gives

$$|\psi_2\rangle = a(j, \lambda) |\lambda\rangle |(1 \oplus j_0) \cdots (1 \oplus j_{m_2})\rangle |\chi(j)\rangle |0\rangle, \quad (153)$$

where

$$\chi(j) = (1 \oplus j_0) \times \cdots \times (1 \oplus j_{m_2}) = \begin{cases} 1 & (j = 0), \\ 0 & (j \neq 0). \end{cases} \quad (154)$$

Next, a Toffoli gate is applied using the first register qubit and first ancilla qubit as controls, and the second ancilla qubit as a target. The state is then

$$|\psi_3\rangle = a(j, \lambda) |\lambda\rangle |(1 \oplus j_0) \cdots (1 \oplus j_{m_2})\rangle |\chi(j)\rangle |\lambda\chi(j)\rangle. \quad (155)$$

The circuit now applies a controlled- Z to the 0-qubit in the second register and the second ancilla qubit giving

$$|\psi_4\rangle = a(j, \lambda) (-1)^{\lambda\chi(j)(1 \oplus j_0)} \times |\lambda\rangle |(1 \oplus j_0) \cdots (1 \oplus j_{m_2})\rangle |\chi(j)\rangle |\lambda\chi(j)\rangle. \quad (156)$$

The circuit now acts to disentangle the ancilla qubits so that they can be safely discarded. Applying the second Toffoli gate gives

$$|\psi_5\rangle = a(j, \lambda) (-1)^{\lambda\chi(j)(1 \oplus j_0)} \times |\lambda\rangle |(1 \oplus j_0) \cdots (1 \oplus j_{m_2})\rangle |\chi(j)\rangle |0\rangle. \quad (157)$$

Applying the second $C_{2M_2-1}(X)$ gate gives

$$|\psi_6\rangle = a(j, \lambda) (-1)^{\lambda\chi(j)(1 \oplus j_0)} \times |\lambda\rangle |(1 \oplus j_0) \cdots (1 \oplus j_{m_2})\rangle |0\rangle |0\rangle. \quad (158)$$

Discarding the ancilla qubits, applying the final X gates, and restoring the suppressed summation symbols gives the final state

$$|\psi_f\rangle = \sum_{\lambda=0}^1 \sum_{j=0}^{2M_2-1} a(j, \lambda) (-1)^{\lambda\chi(j)(1 \oplus j_0)} |\lambda\rangle |j_0 \cdots j_{m_2}\rangle. \quad (159)$$

To see that this is the final state produced by U_0^λ , suppose the first register qubit (that acts as the control for U_0^λ) is in the state $|\lambda = 0\rangle$ so that $a(j, 1) = 0$ for all j . The initial state [Eq. (151)] is then (suppressing the ancilla qubits)

$$|\psi_0\rangle = |\lambda = 0\rangle \sum_{j=0}^{2M_2-1} a(j, 0) |j_0 \cdots j_{m_2}\rangle. \quad (160)$$

The final state [Eq. (159)] in this case is

$$|\psi_f\rangle = |\lambda = 0\rangle \sum_{j=0}^{2M_2-1} a(j, 0) |j_0 \cdots j_{m_2}\rangle. \quad (161)$$

Note that the factor in Eq. (159) that is a power of -1 is equal to 1 since $\lambda = 0$ in the exponent. We see that $|\psi_0\rangle = |\psi_f\rangle$ and so the circuit has applied the identity operation to the second register when the control qubit is in the state $|\lambda = 0\rangle$ as desired. Now suppose that $|\lambda = 1\rangle$ so that $a(j, 0) = 0$ for all j . Suppressing the ancilla qubits again, the initial state is now [Eq. (151)]

$$|\psi_0\rangle = |\lambda = 1\rangle \sum_{j=0}^{2M_2-1} a(j, 1) |j_0 \cdots j_{m_2}\rangle. \quad (162)$$

The final state [Eq. (159)] is now

$$|\psi_f\rangle = |\lambda = 1\rangle \times \otimes \left[-a(0, 1)|0 \cdots 0\rangle + \sum_{j=1}^{2M_2-1} a(j, 1)|j_0 \cdots j_{m_2}\rangle \right]. \quad (163)$$

To compare with Eq. (118), suppose $a(0, 1) = 1$ and $a(j, 1) = 0$ for $j \neq 0$. Then from Eq. (162), $|\psi_0\rangle = |\lambda = 1\rangle|0\rangle$, and from Eq. (163),

$$|\psi_f\rangle = -|\lambda = 1\rangle|0\rangle, \quad (164)$$

in agreement with Eq. (118). Similarly, suppose $a(j, 1) = 1$ and $a(k, 1) = 0$ for $k \neq j$. The initial state is now $|\psi_0\rangle = |\lambda = 1\rangle|j\rangle$, and the final state is

$$|\psi_f\rangle = |\lambda = 1\rangle|j\rangle, \quad (165)$$

in agreement with Eq. (118). Thus, when the control is in the state $|\lambda = 1\rangle$, the circuit applies U_0 to the second register, as desired. We see that the circuit in Fig. 14 does indeed apply a controlled- U_0 operation.

The resource requirements for the controlled- U_0 circuit follows from Fig. 14 and our discussion of U_0 in Sec. IV B. Notice that the circuit in Fig. 14 is nearly identical to the circuit for U_0 that appears in Fig. 9. It has added a one-qubit first register, a second ancilla qubit, and two Toffoli gates. These increase the circuit depth and width, and the number of Toffoli gates, but these increases are independent of ε_d and so will not impact the scaling of resource requirements with ε_d . These requirements result from the part of the circuit (in Fig. 14) that is identical with the U_0 circuit. Thus the circuit depth is [see Eq. (129)]

$$D(U_0^\lambda) = O(\log_2 1/\varepsilon_d). \quad (166)$$

Similarly, the circuit width is [see Eq. (130)]

$$W(U_0^\lambda) = O(\log_2 1/\varepsilon_d), \quad (167)$$

and the number of Toffoli gates is [Eq. (131)]

$$\mathcal{T}(U_0^\lambda) = O(\log_2 1/\varepsilon_d). \quad (168)$$

Finally, from Fig. 14, we see that there are no controlled- ρ gates and so

$$\#(\text{controlled-}\rho)_{U_0^\lambda} = 0. \quad (169)$$

E. Resource requirements

The resources needed to implement $\Lambda(Q)$ follow from Fig. 11 and the results of Sec. IV D. From Fig. 11 we see that implementing $\Lambda(Q)$ requires up to $M_1 - 1$ controlled- Q operations, while the resources needed to apply Q are given in Eqs. (140)–(143).

The depth $D(\Lambda(Q))$ is thus upper-bounded by $(M_1 - 1)D(Q)$. From Eq. (140) and $M_1 = O(1/\varepsilon_1)$, we have

$$D(\Lambda(Q)) = O[(1/\varepsilon_1\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)] + O(1/\varepsilon_1\varepsilon_\gamma). \quad (170)$$

TABLE I. Resource requirements for the three oracles O , Q , and $\Lambda(Q)$ as a function of the error tolerances ε_1 , ε_d , and ε_γ . See the text for further discussion.

Oracle	O	Q	$\Lambda(Q)$
Depth	D_1^a	D_1^a	D_2^b
Width	W_1^c	W_1^c	W_2^d
Toffolis	\mathcal{T}_1^e	\mathcal{T}_1^e	\mathcal{T}_2^f
Controlled- ρ	C_1^g	C_1^g	C_2^h

^a $D_1 = O[(1/\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)] + O(1/\varepsilon_\gamma)$.

^b $D_2 = O[(1/\varepsilon_1\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)] + O(1/\varepsilon_1\varepsilon_\gamma)$.

^c $W_1 = O(\log_2 1/\varepsilon_d) + O(\log_2 1/\varepsilon_\gamma)$

^d $W_2 = O(\log_2 1/\varepsilon_1) + O(\log_2 1/\varepsilon_d) + O(\log_2 1/\varepsilon_\gamma)$.

^e $\mathcal{T}_1 = O[(1/\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)]$.

^f $\mathcal{T}_2 = O[(1/\varepsilon_1\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)]$.

^g $C_1 = O(1/\varepsilon_\gamma)$.

^h $C_2 = O(1/\varepsilon_1\varepsilon_\gamma)$.

From Fig. 11 we see that the circuit width of $\Lambda(Q)$ is $m_1 + W(Q)$. From Eq. (141) and $m_1 = O(\log_2 1/\varepsilon_1)$, we have

$$W(\Lambda(Q)) = O(\log_2 1/\varepsilon_1) + O(\log_2 1/\varepsilon_d) + O(\log_2 1/\varepsilon_\gamma). \quad (171)$$

The number of Toffoli gates is upper-bounded by $(M_1 - 1)\mathcal{T}(Q)$. From Eq. (142) we have

$$\mathcal{T}(\Lambda(Q)) = O[(1/\varepsilon_1\varepsilon_d)(\log_2 1/\varepsilon_d)(\log_2 1/\varepsilon_\gamma)]. \quad (172)$$

Finally, the number of controlled- ρ operations is upper-bounded by $(M_1 - 1)[\#(\text{controlled-}\rho)_Q]$. Using Eq. (143), we have

$$\#(\text{controlled-}\rho)_{\Lambda(Q)} = O(1\varepsilon_1\varepsilon_\gamma). \quad (173)$$

VI. DISCUSSION

(a) In this paper, we have presented quantum circuits that implement the three oracles used in the quantum Navier-Stokes and quantum PDE algorithms of Refs. [2,3], and we determined the resources needed to implement these circuits. Specifically, we determined the circuit depths and widths, as well as the number of non-Clifford gates (Toffoli gates and controlled-small-angle rotations). The resource requirements were given as a function of the error tolerances ε_1 (error in the QAEA estimate of a Riemann sum), ε_d (error in approximating a definite integral by a Riemann sum), and ε_γ (truncation error in approximating an integrand by an m_3 -bit approximation). We collect these costs in Table I. As noted in Sec. I, our goal in this paper has been to obtain circuit implementations for the quantum oracles O , Q , and $\Lambda(Q)$. We leave the task of making these circuits optimal and fault-tolerant to future work.

In Sec. II A 2 we noted that the approximate solution $\alpha(t)$ satisfies an error upper bound [Eq. (16)] with probability $1 - \delta$. The failure probability δ is user-specified. It is related to the probability δ_1 that a run of the QAEA yields an estimate that violates the error upper bound appearing in Eq. (43). The two failure probabilities are related through Eq. (18). As seen in the remarks following Eq. (42), a single run of the QAEA has a failure probability $\delta_1 \sim 0.19$ (when $k = 1$). As noted

there, Heinrichs [17] showed that by running the QAEA M times and returning the median of the M results, the failure probability δ_1 becomes

$$\delta_1 = \exp[-M/8]. \quad (174)$$

In this way, the failure probability δ_1 can be made as small as desired by rerunning the QAEA a sufficient number of times. Having said that, it is actually more useful to solve Eq. (174) for M :

$$M = 8 \lceil \ln(1/\delta_1) \rceil. \quad (175)$$

The reason is that, given a desired failure probability δ , we can determine the necessary δ_1 through Eq. (18). Having δ_1 , Eq. (175) determines the number of times, M , that the QAEA must be rerun so that the QPDE algorithm has failure probability δ .

We see that the parameters ε_1 , ε_d , ε_γ , and δ determine the resource requirements for the oracle circuits presented in this paper. Important next steps are constructing oracle circuits that have reduced resource requirements, and are fault-tolerant.

(b) We close with a few remarks related to quantum speed-up for classical and quantum algorithms containing oracles, in the case in which the quantum oracles have known circuit implementations.

We begin with a few definitions. Suppose the quantum (classical) algorithm uses N_q (N_c) total oracle calls. Suppose further that the quantum algorithm uses k_q oracles O_i^q that take time τ_i^q to execute, with $1 \leq i \leq k_q$. Similarly, let the classical algorithm use k_c oracles O_j^c which execute in time τ_j^c , with $1 \leq j \leq k_c$. Finally, let $\tau_q = \max_i(\tau_i^q)$ and $\tau_c = \max_j(\tau_j^c)$ be the time to execute the slowest quantum and classical oracles, respectively.

When discussing algorithms with oracles, the first level of analysis assumes an oracle call takes a single time step [23]. One algorithm then has a speed-up over another if it uses fewer oracle calls for all instances of the problem of interest. We assume the quantum algorithm has a power-law speed-up over the classical algorithm, $N_q = (N_c)^\alpha$, where $0 < \alpha < 1$. Let T_q (T_c) be the time to execute the quantum (classical) algorithm. We go beyond this first level of analysis and define speed-up to mean $N_q = (N_c)^\alpha$ and $T_q < T_c$.

The time T_c is then

$$T_c = \sum_{j=1}^{k_c} v_j^c \tau_j^c \leq N_c \tau_c, \quad (176)$$

where v_j^c is the number of times oracle O_j^c is used.

Similarly, the time T_q is

$$T_q = \sum_{i=1}^{k_q} v_i^q \tau_i^q \leq N_q \tau_q, \quad (177)$$

where v_i^q is the number of times oracle O_i^q is used. Let C_i denote the quantum circuit for oracle O_i^q , $\mathcal{D}(C_i)$ its circuit depth, and $\mathcal{D} = \max_i(\mathcal{D}(C_i))$ the maximum oracle circuit depth. Let oracle circuit C_{i_*} be the circuit for which $\mathcal{D}(C_{i_*}) = \mathcal{D}$, and let τ_g be the time needed to implement its slowest quantum gate. If more than one circuit satisfies this condition, we pick

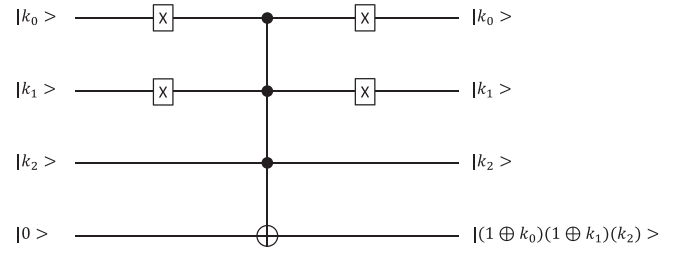


FIG. 15. Quantum circuit implementing the multi-controlled NOT gate $C_j(X)$ for $j = 4$ and $m = 3$. The binary coefficients for $j = 4$ are $j_0 = 0$, $j_1 = 0$, and $j_2 = 1$. This is Fig. 5, except that we have used Pauli X gates to convert open circles to closed circles. As before, the circuit applies X to the target qubit only when $k_0 = 0$, $k_1 = 0$, and $k_2 = 1$ (viz. $|k\rangle = |j\rangle = |4\rangle$), and it acts as the identity for all $|k\rangle \neq |4\rangle$. The gate at the center of the figure is $C_j(X)$ with $j = M - 1$ and $M = 2^m = 8$. It acts as the identity for all CB states $|k\rangle \neq |7\rangle$, and applies X to the target qubit when $|k\rangle = |111\rangle$.

the circuit with the largest τ_g . Then $\tau_q \leq \mathcal{D}\tau_g$ and Eq. (177) becomes

$$T_q \leq N_q \mathcal{D}\tau_g. \quad (178)$$

We can ensure that $T_q < T_c$ if we require $N_q \mathcal{D}\tau_g < T_c$. This leads to a sufficient condition for quantum speed-up,

$$\tau_g < \left(\frac{N_c}{N_q}\right) \frac{\tau_c}{\mathcal{D}} = (N_c)^{1-\alpha} \frac{\tau_c}{\mathcal{D}}, \quad (179)$$

where $0 < \alpha < 1$. Thus, if the time τ_g to run the slowest gate appearing in C_{i_*} satisfies this upper bound, there is a quantum speed-up [$T_q < T_c$ and $N_q = (N_c)^\alpha$]. This requirement on τ_g is a condition placed on the quantum hardware. The RHS upper bound depends on the classical algorithm through N_c ; the classical hardware through τ_c ; the oracle circuit implementations through \mathcal{D} ; and the first level quantum speed-up through the exponent α . We see that the (RHS) upper bound increases with decreasing α , making the speed-up inequality easier to satisfy in this regime. Similarly, the upper bound decreases as the maximum oracle circuit depth \mathcal{D} increases, making speed-up harder to achieve in this regime.

ACKNOWLEDGMENTS

I thank Max Porter for valuable suggestions, and T. Howell III for continued support.

APPENDIX: QUANTUM CIRCUIT FOR THE $C_j(X)$ GATE

Here we show how the $C_j(X)$ gate (Sec. III C) can be implemented using Toffoli gates and Pauli X gates. See also Refs. [24–26].

Referring to Eq. (95), we see that (i) the $C_j(X)$ gate acts on two registers, the first containing $m_2 \rightarrow m$ qubits and the second containing one qubit; (ii) the binary coefficients of j are j_0, \dots, j_{m-1} ; and (iii) those of k are k_0, \dots, k_{m-1} . Figure 5 showed the circuit diagram for $C_j(X)$ when $j = 4$ and $m = 3$. Figure 15 redraws $C_4(X)$ using Pauli X gates to convert open circles to closed circles [27]. Thus implementing $C_j(X)$ reduces to implementing $C_{M-1}(X)$, where $M = 2^m$. In the case of Fig. 15, $M - 1 = 7$.

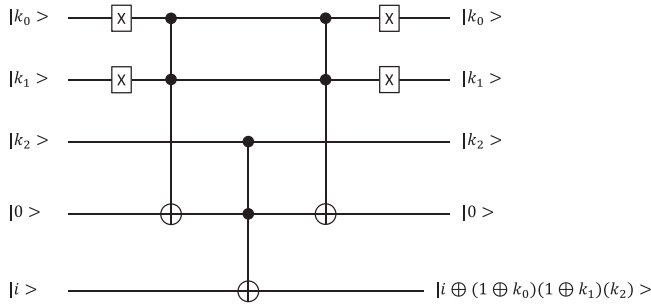


FIG. 16. Implementing the action of the multi-controlled NOT gate $C_j(X)$ for $j = 4$ and $m = 3$ using Toffoli gates and Pauli X gates. The net effect of the three Toffoli gates (and the discarded ancilla qubit) is to apply $C_{M-1}(X)$ with $M - 1 = 7$. As before, the circuit applies X to the target qubit only when $k_0 = 0$, $k_1 = 0$, and $k_2 = 1$ (viz. $|k\rangle = |j\rangle = |4\rangle$), and acts as the identity for all $|k\rangle \neq |4\rangle$.

The basic ingredient for implementing $C_{M-1}(X)$ is the Toffoli gate U_{Tof} , whose action is well-known:

$$U_{\text{Tof}}|k_0\rangle \otimes |k_1\rangle \otimes |i\rangle = |k_0\rangle \otimes |k_1\rangle \otimes |i \oplus k_0k_1\rangle. \quad (\text{A1})$$

The Toffoli gate multiplies the control inputs k_0 and k_1 and adds their product to the target qubit input i using binary addition. Note that when $i = 0$, the final target state $|k_0k_1\rangle$ stores the product of the control inputs in the label of the target final state. We utilize this in the circuit implementation of $C_{M-1}(X)$.

Suppose there are m control inputs k_0, \dots, k_{m-1} whose product we want to add to the target input i . We use Toffoli gates to build up the product $k_0 \cdots k_{m-1}$ iteratively. The first Toffoli gate acts on control qubits 0 and 1 and uses an ancilla qubit in the initial state $|0\rangle$ as the target. The Toffoli gate leaves the ancilla in the state $|k_0k_1\rangle$ as explained in the previous paragraph. The second Toffoli gate uses control qubit 2 and the first ancilla as control qubits, and a second ancilla

in the state $|0\rangle$ as the target. The second Toffoli gate leaves the second ancilla in the state $|k_0k_1k_2\rangle$. Repeating this process $m - 2$ times leaves ancilla $m - 2$ in the state $|k_0 \cdots k_{m-2}\rangle$. To complete constructing the product, we apply another Toffoli gate using control qubit $m - 1$ and ancilla $m - 2$ as controls and the single qubit in the original second register whose initial state is $|i\rangle$ as the target qubit. The final state of the second register qubit is then $|i \oplus k_0 \cdots k_{m-1}\rangle$. It is important to note that at this point the $m - 2$ ancilla qubits are entangled with the original two registers. To disentangle them, we must reapply the first $m - 2$ Toffoli gates in reverse order. Once this is done, we discard the unentangled ancilla qubits. The original two registers are left in the state $|k\rangle_m \otimes |i \oplus k_0 \cdots k_{m-1}\rangle_1$, which is the desired state on the right-hand side of Eq. (95).

Before illustrating this circuit construction with an example, we determine the resources used. The number of Toffoli gates used when $m \geq 2$ is

$$\mathcal{T} = (m - 2) + 1 + (m - 2) = 2m - 3, \quad (\text{A2})$$

and the number of ancilla qubits used is

$$\mathcal{A} = m - 2. \quad (\text{A3})$$

The circuit depth is controlled by the number of Toffoli gates, and so

$$D(C_j(X)) = 2m - 3 = O(\log_2 M), \quad (\text{A4})$$

and the circuit width (including ancilla qubits) is

$$W(C_j(X)) = m + 1 + (m - 2) = O(\log_2 M). \quad (\text{A5})$$

To illustrate this construction, Fig. 16 shows how to implement $C_j(X)$ for $j = 4$, $m = 3$. As seen before, $j_0 = 0$, $j_1 = 0$, and $j_2 = 1$. We see that the circuit uses $2m - 3 = 3$ Toffoli gates and $m - 2 = 1$ ancilla qubit. We leave it as an exercise to implement $C_j(X)$ for $j = 4$ and $m = 4$. This will require five Toffoli gates, two ancilla qubits, and six Pauli X gates.

- [1] R. P. Feynman, Simulating physics with computers, *Int. J. Theor. Phys.* **21**, 467 (1982).
- [2] F. Gaitan, Finding flows of a Navier-Stokes fluid through quantum computing, *npj Quantum Inf.* **6**, 61 (2020).
- [3] F. Gaitan, Finding solutions of the Navier-Stokes equations through quantum computing - recent progress, a generalization, and next steps forward, *Adv. Quantum Tech.* **4**, 2100055 (2021).
- [4] J. Yepez, Lattice-gas quantum computation, *Int. J. Mod. Phys. C* **9**, 1587 (1998).
- [5] J. Yepez, Quantum computation of fluid dynamics, in *Quantum Computing and Quantum Communications*, Lecture Notes in Computer Science Vol. 1509, edited by C. P. Williams (Springer, New York, 1999), p. 34.
- [6] J. Yepez, Quantum lattice-gas model for computational fluid dynamics, *Phys. Rev. E* **63**, 046702 (2001).
- [7] J. Yepez, Quantum lattice-gas model for Burgers equation, *J. Stat. Phys.* **107**, 203 (2002).
- [8] R. Steijl, Quantum algorithms for fluid simulations, in *Advances in Quantum Communication and Information*, edited by F. Bulnes, V. N. Stavrou, O. Mozorov, A. V. Bourdine (Intech-Open, 2019).
- [9] R. Steijl and G. N. Barakos, Parallel evaluation of quantum algorithms for computational fluid dynamics, *Comput. Fluids* **173**, 22 (2018).
- [10] G. Xu, A. J. Daley, P. Givi, and R. D. Somma, Turbulent mixing simulation via a quantum algorithm, *AIAA J.* **56**, 687 (2018).
- [11] Y. Cao, A. Papageorgiou, I. Petras, J. Traub, and S. Kais, Quantum algorithm and circuit design solving the Poisson equation, *New J. Phys.* **15**, 013021 (2013).
- [12] J. Arrazola, T. Kalajdzievski, C. Weedbrook, and S. Lloyd, Quantum algorithm for nonhomogeneous linear partial differential equations, *Phys. Rev. A* **100**, 032306 (2019).
- [13] F. Oz, R. K. S. S. Vupula, K. Kara, and F. Gaitan, Solving Burgers equation with quantum computing, *Quantum Inf. Process.* **21**, 30 (2022).
- [14] A. Kitaev, Quantum measurements and the Abelian stabilizer problem, [arXiv:quant-ph/9511026](https://arxiv.org/abs/quant-ph/9511026) (1995).
- [15] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, Quantum amplitude amplification and estimation, *AMS Contemp. Math.* **305**, 53 (2002).
- [16] E. Novak, Quantum complexity of integration, *J. Complexity* **17**, 2 (2001).

- [17] S. Heinrichs, Quantum summation with an application to integration, *J. Complexity* **18**, 1 (2002).
- [18] B. Kacwicz, Almost optimal solution of initial-value problems by randomized and quantum algorithms, *J. Complexity* **22**, 676 (2006).
- [19] W. Gautschi, *Numerical Analysis* (Birkhäuser, New York, 2012).
- [20] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations* (Cambridge University Press, New York, 2009).
- [21] L. C. Evans, *Partial Differential Equations* (American Mathematical Society, Providence, RI, 1998).
- [22] R. Courant, K. O. Friedrichs, and H. Lewy, On the partial difference equations of mathematical physics, *Math. Ann.* **100**, 32 (1928); *IBM J. Res. Dev.* **11**, 215 (1967).
- [23] C. H. Papadimitriou, *Computational Complexity* (Pearson, New York, 1993).
- [24] L. Isenhower, M. Saffman, and K. Molmer, Multibit C_k NOT gates via Rydberg blockade, *Quantum Inf. Proc.* **10**, 755 (2011).
- [25] S. E. Rasmussen, K. Groenland, R. Gerritsma, K. Schoutens, and N. T. Zinner, Single-step implementation of high-fidelity n-bit Toffoli gate, *Phys. Rev. A* **101**, 022308 (2020).
- [26] A. M. Chen, S. Y. Cho, and M. D. Kim, Implementation of three-qubit Toffoli gate in a single step, *Phys. Rev. A* **85**, 032326 (2012).
- [27] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge, New York, 2000), pp. 184–185.