Representation of the fermionic boundary operator

Ismail Yunus Akhalwaya,[1,2,*] Yang-Hui He,[3,4,5,6,†] Lior Horesh,[7,‡] Vishnu Jejjala,[8,§]
William Kirby,[9,10,∥] Kugendran Naidoo,[8,¶] and Shashanka Ubaru[7,#]

[1]*IBM Research Africa, Johannesburg 2000, South Africa*
[2]*School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, WITS 2050, South Africa*
[3]*London Institute for Mathematical Sciences, Royal Institution, London W1S 4BS, United Kingdom*
[4]*Department of Mathematics, City, University of London, London EC1V 0HB, United Kingdom*
[5]*Merton College, University of Oxford, Oxford OX1 4JD, United Kingdom*
[6]*School of Physics, NanKai University, Tianjin 300071, China*
[7]*Mathematics of AI, IBM Research, T. J. Watson Research Center, Yorktown Heights, New York 10598, USA*
[8]*Mandelstam Institute for Theoretical Physics, School of Physics, NITheCS, and CoE-MaSS, University of the Witwatersrand, Johannesburg, WITS 2050, South Africa*
[9]*Department of Physics and Astronomy, Tufts University, Medford, Massachusetts 02155, USA*
[10]*IBM Quantum, IBM Research, T. J. Watson Research Center, Yorktown Heights, New York 10598, USA*

(Received 24 April 2022; accepted 27 July 2022; published 10 August 2022)

The boundary operator is a linear operator that acts on a collection of high-dimensional binary points (simplices) and maps them to their boundaries. This boundary map is one of the key components in numerous applications, including differential equations, machine learning, computational geometry, machine vision, and control systems. We consider the problem of representing the full boundary operator on a quantum computer. We first prove that the boundary operator has a special structure in the form of a complete sum of fermionic creation and annihilation operators. We then use the fact that these operators pairwise anticommute to produce an $O(n)$-depth circuit that exactly implements the boundary operator without any Trotterization or Taylor-series approximation errors. Having fewer errors reduces the number of shots required to obtain desired accuracies.

DOI: 10.1103/PhysRevA.106.022407

I. INTRODUCTION

Quantum computers are capable of performing certain linear algebraic operations in exponentially large spaces and promise to achieve significant asymptotic speedups over classical computers [1,2]. In recent years, several quantum algorithms have been proposed to leverage the potential of quantum computing [3–8]. These algorithms achieve polynomial to exponential speedups over the best-known classical methods. However, most of these methods require large-scale fault-tolerant quantum computers in order to challenge the classical methods in practice.

The realization of fault tolerance in quantum computers is likely at least several years away. Present-day quantum computers are referred to as noisy intermediate-scale quantum (NISQ) [9] devices and are too small to implement error correction but too large to simulate classically [10]. Development of algorithms that achieve quantum advantage for useful tasks on NISQ devices is a critical next step for the field [9–12].

Boundary operators (also known as boundary maps) are among the most important computational primitives used for the representation and analysis of differential equations [13–15], finite-element methods [16,17], graph and network analysis [18–20], computational geometry [21,22], machine vision [23,24], control systems [25,26], and more. They are also used to bridge the gap between discrete representations, such as graphs and simplicial complexes, and continuous representations, such as vector spaces and manifolds. The graph Laplacians (including higher-order combinatorial Laplacians) can be constructed using the boundary operator [27,28]. Laplacians of graphs and hypergraphs play an important role in spectral clustering [29], a computationally tractable solution to the graph partitioning problem that is prevalent in applications such as image segmentation [30], collaborative recommendation [31], text categorization [32], and manifold learning [33]. Finally, graph Laplacians are crucial in graph neural network architectures [34–36], a recent set of neural network techniques for learning graph representations.

The boundary operator also plays a critical role in topological data analysis (TDA), as a linear operator that acts on a given set of simplices and maps them to their boundaries. TDA is a powerful machine learning and data analysis technique used to extract shape-related information of large data

*IsmailA@za.ibm.com
†hey@maths.ox.ac.uk
‡lhoresh@us.ibm.com
§vishnu@neo.phys.wits.ac.za
∥william.kirby@ibm.com
¶9006597F@students.wits.ac.za
#Shashanka.Ubaru@ibm.com

sets [22,37–39]. TDA permits representing large volumes of data using a few global and interpretable features called Betti numbers [37]. Classical algorithms for TDA and Betti-number calculations are typically computationally expensive. In [7], Lloyd *et al.* proposed a quantum algorithm for TDA, and this algorithm provably achieves exponential speedup over classical TDA algorithms under certain conditions, as shown in [40]. However, their quantum TDA (or QTDA) algorithm requires fault tolerance due to the need for a full quantum phase estimation (made worse by the embedded and repeated use of Grover's search). More recently, Ubaru *et al.* [41] devised a QTDA algorithm that is more amenable to NISQ implementation. The algorithm requires only an $O(n)$-depth quantum circuit and thus has the potential to be an early useful NISQ algorithm that provably (up to weak assumptions) achieves exponential speedup.

In [41], the (generalized) boundary operator B is represented in a novel tensor form, as a complete sum of fermionic creation and annihilation operators.[1] Then, the standard technique of applying a Hermitian operator on a quantum computer is implemented, namely, executing the time-evolution unitary $e^{-\mathtt{i}tB}$ for short time t and "solving" for the second term of the Taylor series $(-\mathtt{i}tB)$. The time evolution is approximated by Trotterization [43–45]. Using this technique, with certain gate cancellations, an $O(n)$-depth circuit representation can be obtained for the boundary operator [41]. However, both the Taylor-series expansion and Trotterization steps accrue errors and require many shots to accurately simulate B.

In this paper, we first present a proof of correctness of the fermionic representation of the boundary operator given in [41,42]. Next, we use a technique developed in [10,46] (where it is called *unitary partitioning*) to exactly express the full Hermitian boundary operator as a unitary operator that has an efficient $O(n)$-depth construction in terms of quantum computing primitives. This short depth circuit, without evolution and Trotterization, analytically implements the boundary operator, allowing for far fewer measurement shots than the standard Hermitian-evolution technique (see Sec. IV). This can be instrumental in many downstream applications such as QTDA [7,41], cohomology problems [42], quantum algorithms for finite-element methods [17], solving partial differential equations [14,15,47], and potential quantum algorithms for machine vision and control systems.

II. THE BOUNDARY OPERATOR AND ITS FERMIONIC REPRESENTATION

In this section, we first introduce the concept of boundary operators in the context of computational geometry and TDA. We then discuss the fermionic creation and annihilation operators and present a fermionic representation for the boundary operator along with a proof establishing its correctness.

[1]In [42], the connection between supersymmetric many-body systems and the (co)homology problem is discussed, and independent of our work, the same fermionic annihilation operator representation for the (co)boundary map is introduced.

A. Computational geometry

In [7], Lloyd *et al.* introduced a quantum algorithm for topological data analysis (QTDA). Simplices are represented by strings of n bits. Each bit corresponds to a vertex, with zero indicating exclusion and one indicating inclusion. Hence, on the quantum computer, the usual computational basis directly maps to simplices. There are 2^n computational basis vectors, one for each unique n-bit binary string, written $|s_k\rangle$, where k indicates the number of vertices in the simplex (i.e., the number of ones in the binary string).[2] In general, the quantum state vector (of length 2^n) can be in a superposition of these basis vectors. Core to the QTDA algorithm is the restricted boundary operator, defined by its action on k-dimensional simplices (among n vertices, hence the superscript (n) in the following):

$$\partial_k^{(n)}|s_k\rangle = \sum_l (-1)^l |s_{k-1}(l)\rangle, \tag{1}$$

where $|s_k\rangle$ represents a simplex and $|s_{k-1}(l)\rangle$ is the simplex of one dimension less than $|s_k\rangle$ with the same vertex set (containing n vertices) but leaving out the lth vertex counting from the left.

1. Fleshing out the restricted boundary operator

It helps to rewrite (1) without the use of index l, which hides some algorithmic steps, because l presupposes that the locations of the ones are known (i.e., which vertices are *in* a given simplex, e.g., $l = 0$ refers to the first 1 in the string reading from left to right). We rewrite using index i:

$$\partial_k^{(n)}|s_k\rangle = \sum_{i=0}^{n-1} \delta_{s_k[i],1}(-1)^{\sum_{j=i+1}^{n-1} s_k[j]}|s_{k-1}(i)\rangle, \tag{2}$$

where we now choose to locate each bit counting from the right starting with index 0. Here we introduce the notation $s_k[i]$ to represent the ith bit of the string. We keep $|s_{k-1}(i)\rangle$ to mean what Lloyd *et al.* meant (except now i is any bit index), namely, $|s_k\rangle$ with the ith vertex set to zero. Crucially, all references to l have been replaced, including the implicitly required knowledge of the location of the last 1 (the sum across i simply runs from 0 to $n-1$). In the Appendix, we illustrate this definition with explicit examples.

Now, $\partial_k^{(n)}$'s action on a single simplex can be split into two cases depending on whether the last vertex, indexed $n-1$, is in the simplex or not. The absence or presence of this last vertex is represented by the bit $s_k[n-1]$ being 0 or 1, respectively. If $s_k[n-1] = 0$, the quantum state corresponding to this single simplex (call it $|s_{k,0}\rangle$) is a single computational basis state whose binary string has exactly k ones somewhere in it, except that the leftmost bit is zero. If we write this basis state as an exponentially long vector, it consists of a column of 2^{n-1} bits, only one of which is 1 (at a location whose binary string consists of those k ones), followed by 2^{n-1} zeros. This

[2]There is a difference in convention for how to index the bits of the binary string. In this paper, we try to be explicit and refer to "counting from the left" or "right." For our introduced indices, we always index from zero.

is similarly the case for $|s_{k,1}\rangle$, but with the single 1 appearing in the second half of the column vector.

If $|s_k\rangle = |s_{k,0}\rangle$ for $k \leqslant n-1$,

$$\begin{aligned}\partial_k^{(n)}|s_{k,0}\rangle &= \sum_{i=0}^{n-1} \delta_{s_k[i],1}(-1)^{\sum_{j=i+1}^{n-1} s_k[j]}|s_{k-1}(i)\rangle \\ &= \sum_{i=0}^{n-2} \delta_{s_k[i],1}(-1)^{\sum_{j=i+1}^{n-2} s_k[j]}|s_{k-1}(i)\rangle \\ &= \begin{pmatrix} \partial_k^{(n-1)} & 0 \\ 0 & 0 \end{pmatrix}|s_{k,0}\rangle. \end{aligned} \tag{3}$$

If $|s_k\rangle = |s_{k,1}\rangle$ for $k \leqslant n$,

$$\begin{aligned}\partial_k^{(n)}|s_{k,1}\rangle &= \sum_{i=0}^{n-1} \delta_{s_k[i],1}(-1)^{\sum_{j=i+1}^{n-1} s_k[j]}|s_{k-1}(i)\rangle \\ &= |s_{k-1}(n-1)\rangle + (-1)\sum_{i=0}^{n-2} \\ &\quad \times\ \delta_{s_k[i],1}(-1)^{\sum_{j=i+1}^{n-2} s_k[j]}|s_{k-1}(i)\rangle \\ &= \begin{pmatrix} 0 & P_{k-1}^{(n-1)} \\ 0 & 0 \end{pmatrix}|s_{k,1}\rangle - 1\begin{pmatrix} 0 & 0 \\ 0 & \partial_{k-1}^{(n-1)} \end{pmatrix}|s_{k,1}\rangle, \end{aligned} \tag{4}$$

where $P_{k-1}^{(n-1)}$ is the projection onto the computational basis states of $n-1$ qubits whose binary strings contain exactly $k-1$ ones. See how the block-diagonal notation allows us to remove or leave alone the kth one precisely in the nth position. In this particular equation, since we act on only $|s_{k,1}\rangle$ and remove the nth vertex, the projection could be replaced by the identity operator.

B. Fermionic boundary operator

Fermionic fields obey Fermi-Dirac statistics, which means that they admit a mode expansion in terms of creation and annihilation oscillators that anticommute. Exploiting this fact, it is convenient to map Pauli spin operators to fermionic creation and annihilation operators. The Jordan-Wigner transformation [48] is one such mapping. In this section, we will make use of it to express the boundary matrix.

The Lloyd *et al.* restricted boundary operator given in (1) is not in a form that can be easily executed on a quantum computer, nor does it act on all orders k at the same time. In particular, it is a high-level description of the action of the boundary operator on a single generic k-dimensional simplex with the location of the ones assumed to be known. Ubaru *et al.* [41] proposed a novel representation that realizes the full boundary operator as a matrix. Furthermore, this representation is in tensor-product form composed of quantum computing primitives that directly map to quantum gates in the quantum circuit model. To begin, define the operator

$$Q^+ := \tfrac{1}{2}(\sigma_x + i\sigma_y) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}. \tag{5}$$

This allowed Ubaru *et al.* to suggest writing the *full* boundary operator in terms of the above operator:

$$\begin{aligned}\partial^{(n)} &:= \sigma_z \otimes \ldots \otimes \sigma_z \otimes Q^+ \\ &\quad + \sigma_z \otimes \ldots \otimes \sigma_z \otimes Q^+ \otimes I \\ &\quad \vdots \\ &\quad + \sigma_z \otimes Q^+ \otimes I \otimes \ldots \\ &\quad + Q^+ \otimes I \otimes I \otimes \ldots \\ &= \sum_{i=0}^{n-1} a_i, \end{aligned} \tag{6}$$

where a_i are the Jordan-Wigner [48] Pauli embeddings corresponding to the n-spin fermionic annihilation operators, hence the title of this paper.[3] To be explicit, a_i is the antisymmetric annihilation operator on mode or orbital i (indexed from the right, starting at zero):

$$\begin{aligned} a_i &:= \underbrace{\sigma_z \otimes \cdots \otimes \sigma_z}_{n-(i+1)} \otimes Q^+ \otimes \underbrace{I \otimes \cdots \otimes I}_{i} \\ &= \sigma_z^{\otimes[n-(i+1)]} \otimes Q^+ \otimes I^{\otimes i}, \end{aligned} \tag{7}$$

$$\{a_i, a_j^\dagger\} = \delta_{ij} I^{\otimes n}. \tag{8}$$

When (6) is realized in matrix form, where the rows' and columns' integer indices (counting from left to right and top to bottom, beginning at zero) are translated into binary, the meaning of these strings corresponds exactly to the simplex strings introduced above. Recall that a one at bit i of the simplex string (counting from the right, starting from zero) corresponds to selecting the ith vertex.

The first key result we need to show is as follows:

Theorem 1. The full boundary operator as defined in (6) is a sum over the restricted boundary operators over simplices of each dimension as defined in Sec. II A 1 as follows:

$$\partial^{(n)} = \sum_{k=1}^{n} \partial_k^{(n)}. \tag{9}$$

Remark 1. A simple sum (as opposed to a direct sum) can be used because the Lloyd *et al.*'s definition of ∂_k has already been extended to a common vertex space of size n. The rewriting constructs the full boundary operator in one step, and the individual ∂_k do not feature at all. Indeed to recover the ∂_k one would need to project into the appropriate space, which is a required step in implementing QTDA on quantum computers [41]. Index i of the a_i fermionic operators indicates which qubit out of n is acted upon by Q^+ (counting from the right) and has nothing to do with ∂_k (and is why there is no k dependence).

[3]While this suggests that only the Jordan-Wigner mapping of fermionic operators does the job, it does open the question of whether other embeddings (e.g., Bravyi-Kitaev [49]) could also play a role, perhaps via translation between the different embeddings. More tantalizingly, since we have recast simplicial homology in terms of these fermionic operators, it would be interesting to explore other geometric structures under this light.

Proof. The proof proceeds by induction, so we will first set up a recurrence relation.

a. Recurrence relation. Equation (6) is amenable to efficient quantum circuit construction, as we show in Sec. III. However, first, we need to prove that Eq. (6) is valid and correctly implements Lloyd *et al.*'s high-level description (9). The easiest way to prove Eq. (9) is by noticing that the left-hand side satisfies the following recurrence relation and then connecting the recurrence relation to the right-hand side of (9):

$$\partial^{(n)} = Q^{+} \otimes I^{\otimes(n-1)} + \sigma_z \otimes \partial^{(n-1)},$$
$$\partial^{(1)} = Q^{+}. \tag{10}$$

It helps to write (10) in block-diagonal form,

$$\partial^{(n)} = \begin{pmatrix} +\partial^{(n-1)} & I^{\otimes(n-1)} \\ 0 & -\partial^{(n-1)} \end{pmatrix}, \tag{11}$$

and think about the action of $\partial^{(n)}$ in terms of $\partial^{(n-1)}$.

The operator $\partial^{(n-1)}$ acts on vectors of size 2^{n-1}, while $\partial^{(n)}$ acts on vectors of size 2^n. The vector of size 2^n can be seen as two halves of size 2^{n-1}. The top half consists of simplices where the nth vertex is not in the simplex. The bottom half corresponds to a "copy" of the $n-1$ space, but now with the nth vertex selected. For example, edges in the upper half become triangles in the lower half since the nth vertex is added. The top left block of $\partial^{(n)}$ in (11) is $\partial^{(n-1)}$, acting on the "top half" simplices, where the nth vertex is not selected ($s_k[n-1]=0$ and k has to be strictly less than n) and returns simplices (the boundaries) in the top half (since taking the boundary can never add the nth vertex). The top right block $I^{\otimes(n-1)}$ acts on simplices in the bottom half ($s_k[n-1]=1$) and returns simplices in the top half, which corresponds to simply removing the nth vertex and leaving everything else the same (hence the identity operator). Finally, the bottom right block acts on simplices in the bottom half ($s_k[n-1]=1$) and returns simplices in the bottom half, corresponding to those returned by taking the boundary operator acting on the first $n-1$ vertices and *leaving* the nth vertex selected. The bottom returned vertices need to be multiplied by -1 because acting on the first $n-1$ vertices by $\partial^{(n-1)}$ does not factor in the extra -1 from the nth vertex in definition (2).

b. Inductive proof of simplex action. We now prove (9) via induction. The base case, $\partial^{(1)} = \partial_1^{(1)}$, clearly implements the Lloyd *et al.* definition because it takes the single vertex vector to the null vector as required. Now we assume $\partial^{(n)}$ for $n \geqslant 1$ implements the sum of restricted boundary operators (9) up to n and use that to show that ∂_{n+1} correctly implements the sum up to $n+1$, i.e., that

$$\partial^{(n+1)} = \sum_{k=1}^{n+1} \partial_k^{(n+1)}. \tag{12}$$

From (11), the left-hand side of (12) is

$$\partial^{(n+1)} = \begin{pmatrix} \partial^{(n)} & I^{\otimes(n)} \\ 0 & -\partial^{(n)} \end{pmatrix}, \tag{13}$$

and from the inductive assumption (9), this becomes

$$\partial^{(n+1)} = \begin{pmatrix} \sum_{k=1}^{n} \partial_k^{(n)} & I^{\otimes n} \\ 0 & -\sum_{k=1}^{n} \partial_k^{(n)} \end{pmatrix}. \tag{14}$$

The right-hand side of (12) contains terms of the form $\partial_k^{(n+1)}$. All $\partial_k^{(n+1)}$ act on $n+1$ vertices, so in matrix form they are all of the same dimension. From (3) and (4) we have

$$\partial_k^{(n+1)}|s_{k,0}\rangle = \begin{pmatrix} \partial_k^{(n)} & 0 \\ 0 & 0 \end{pmatrix}|s_{k,0}\rangle, \quad 1 \leqslant k \leqslant n, \tag{15}$$

$$\partial_k^{(n+1)}|s_{k,1}\rangle = \begin{pmatrix} 0 & P_{k-1}^{(n)} \\ 0 & -\partial_{k-1}^{(n)} \end{pmatrix}|s_{k,1}\rangle, \quad 1 \leqslant k \leqslant n+1. \tag{16}$$

Now since any $|s\rangle = \sum_k |s_k\rangle = \sum_k(|s_{k,0}\rangle + |s_{k,1}\rangle)$ and the matrix in (15) takes $|s_{k,1}\rangle$ to the null vector, $\begin{pmatrix} \partial_k^{(n)} & 0 \\ 0 & 0 \end{pmatrix}|s_{k,1}\rangle = 0$, while the matrix in (16) similarly takes $|s_{k,0}\rangle$ to the null vector, $\begin{pmatrix} 0 & P_{k-1}^{(n)} \\ 0 & -\partial_{k-1}^{(n)} \end{pmatrix}|s_{k,0}\rangle = 0$, we can combine (15) and (16) while paying attention to different values of k:

$$\partial_1^{(n+1)} = \begin{pmatrix} \partial_1^{(n)} & P_0^{(n)} \\ 0 & 0 \end{pmatrix}, \quad k = 1, \tag{17}$$

$$\partial_k^{(n+1)} = \begin{pmatrix} \partial_k^{(n)} & P_{k-1}^{(n)} \\ 0 & -\partial_{k-1}^{(n)} \end{pmatrix}, \quad 2 \leqslant k \leqslant n, \tag{18}$$

$$\partial_{n+1}^{(n+1)} = \begin{pmatrix} 0 & P_n^{(n)} \\ 0 & -\partial_n^{(n)} \end{pmatrix}, \quad k = n+1, \tag{19}$$

where $P_0^{(n)} = |0\cdots0\rangle\langle0\cdots0|$ (n zeros) and $P_n^{(n)} = |1\cdots1\rangle\langle1\cdots1|$ (n ones). Note that $\partial_{n+1}^{(n+1)}$ produces no top left block and $\partial_1^{(n+1)}$ produces no bottom right block.

Therefore, the right-hand side of (12) is

$$\sum_{k=1}^{n+1} \partial_k^{(n+1)} = \begin{pmatrix} \sum_{k=1}^{n} \partial_k^{(n)} & I^{\otimes n} \\ 0 & -\sum_{k=1}^{n} \partial_k^{(n)} \end{pmatrix} \tag{20}$$

since $\sum_{k=0}^{n} P_k^{(n)} = I^{\otimes n}$. Therefore, the right-hand side is equal to the left-hand side (14). ■

With the proof in hand, we now understand why (6) and its recursive form (10) can produce (1) only in summation form (9). In particular, ∂_k on a simplex with one more potential vertex $(n+1)$ is related to both ∂_k and ∂_{k-1} on the original number of vertices (n). Therefore, the full chain of sums is needed to build up a recursive construction, where we sequentially add one vertex at a time, which then makes a quantum-implementable tensor definition possible.

III. UNITARY CIRCUIT

The boundary operator above is not unitary. It is also not Hermitian, which makes it more difficult to use known techniques to implement it on a quantum computer. As a first step we consider a Hermitian version of the boundary operator constructed by simply adding the Hermitian conjugate. It turns out that this Hermitian version happens to be a scaled unitary operator, therefore requiring no further work to obtain a unitary circuit. Naturally, the Hermitian version is not identical to the non-Hermitian version, but a simple projection-based reconstruction is discussed below.

Every Q^+ in the tensor product now becomes $\sigma_x = Q^+ + (Q^+)^\dagger$. Hence, the full Hermitian boundary operator is

$$\begin{aligned} B = \partial + \partial^\dagger &= \sigma_z \otimes \ldots \otimes \sigma_z \otimes \sigma_x \\ &+ \sigma_z \otimes \ldots \otimes \sigma_z \otimes \sigma_x \otimes I \\ &\vdots \\ &+ \sigma_z \otimes \sigma_x \otimes I \otimes \ldots \\ &+ \sigma_x \otimes I \otimes I \otimes \ldots \\ &= \sum_{i=0}^{n-1} Q_i, \end{aligned} \tag{21}$$

where I denotes the single-qubit identity operator and

$$Q_i := a_i + a_i^\dagger, \tag{22}$$

with a_i as defined in (7). This Q_i is a Kronecker product of n Pauli matrices, of which the $n - (i + 1)$ leftmost ones are σ_z. The goal is now to construct a unitary circuit to implement either B or $\exp(\mathrm{i}Bt)$ (depending on the application). Towards this end, it is extremely useful to note the following:

Lemma 1. Q_i pairwise anticommute: $\{Q_i, Q_j\} = 0$ for $i \neq j$.

Proof. The mixed-product identity for the Kronecker product is

$$(A \otimes B) \cdot (C \otimes D) = (AC) \otimes (BD). \tag{23}$$

Hence, we have (assuming without loss of generality that $i > j$ so that there are more σ_z matrices in Q_j)

$$\begin{aligned} \{Q_i, Q_j\} = \{&\underbrace{\sigma_z \otimes \cdots \otimes \sigma_z}_{n-(i+1)} \otimes \sigma_x \otimes \underbrace{I \otimes \cdots \otimes I}_{i}, \\ &\times \underbrace{\sigma_z \otimes \cdots \otimes \sigma_z}_{n-(j+1)} \otimes \sigma_x \otimes \underbrace{I \otimes \cdots \otimes I}_{j}\}. \end{aligned} \tag{24}$$

Since $\{\sigma_\ell, \sigma_k\} = 2\delta_{\ell k}\mathbf{I}$ and $[\sigma_k, I] = 0$ for any $k = x, y, z$, (24) immediately yields the result. ■

We now use the fact that any real linear combination of pairwise anticommuting Pauli operators is unitarily equivalent to a single Pauli operator, up to some rescaling [46]. One can think of this as a generalization of the Bloch sphere to more than three Pauli operators. Moreover, the unitary that maps the linear combination to the single Pauli operator can be efficiently constructed. This technique was developed for the purpose of reducing the number of distinct terms in Hamiltonians to be simulated using variational quantum eigensolvers, in which context it is known as *unitary partitioning* [10,46,50].

For neighboring Q, Q_{i-1} and Q_i,

$$\begin{aligned} -\mathrm{i}Q_{i-1}Q_i &= -\mathrm{i}(\underbrace{\sigma_z \otimes \cdots \otimes \sigma_z}_{n-i} \otimes \sigma_x \otimes \underbrace{I \otimes \ldots \otimes I}_{i-1}) \\ &\quad \times \cdot(\underbrace{\sigma_z \otimes \cdots \otimes \sigma_z}_{n-(i+1)} \otimes \sigma_x \otimes \underbrace{I \otimes \ldots \otimes I}_{i}) \\ &= -\mathrm{i}I^{\otimes[n-(i+1)]} \otimes \sigma_z\sigma_x \otimes \sigma_x \otimes I^{\otimes(i-1)} \\ &= -\mathrm{i}I^{\otimes[n-(i+1)]} \otimes \mathrm{i}\sigma_y \otimes \sigma_x \otimes I^{\otimes(i-1)} \\ &= Y_i X_{i-1}, \end{aligned} \tag{25}$$

where X_{i-1} and Y_i are, respectively,

$$X_{i-1} = I^{\otimes(n-i)} \otimes \sigma_x \otimes I^{\otimes(i-1)}, \tag{26}$$

$$Y_i = I^{\otimes[n-(i+1)]} \otimes \sigma_y \otimes I^{\otimes i}. \tag{27}$$

Hence, $-\mathrm{i}Q_{i-1}Q_i$ is itself a Pauli operator and commutes with all Pauli terms in B except for Q_{i-1} and Q_i. Therefore, a rotation generated by $-\mathrm{i}Q_{i-1}Q_i$ affects only those two terms. For an arbitrary linear combination $\alpha Q_i + \beta Q_{i-1}$ for real α and β, we define the following rotation generated by $-\mathrm{i}Q_{i-1}Q_i$:

$$\begin{aligned} R_i &\equiv \exp\left(\frac{Q_{i-1}Q_i}{2}\mathrm{atan2}(\alpha, \beta)\right) \\ &= \exp\left(\frac{\mathrm{i}Y_iX_{i-1}}{2}\mathrm{atan2}(\alpha, \beta)\right). \end{aligned} \tag{28}$$

It can be shown that the adjoint action of R_i on the linear combination is

$$R_i(\alpha Q_i + \beta Q_{i-1})R_i^\dagger = \sqrt{\alpha^2 + \beta^2}Q_{i-1}. \tag{29}$$

For details, we refer the reader to the unitary partitioning papers [10,46,50].

Therefore, we can map B, which, as given in (21), is the sum of all Q_i, to a single Q_i via a sequence of such rotations as follows:

$$\begin{aligned} B = \sum_{i=0}^{n-1} Q_i &\xrightarrow{R_{n-1} \text{ with } \alpha = 1, \beta = 1} \sqrt{2}Q_{n-2} + \sum_{i=0}^{n-3} Q_i \\ &\times \xrightarrow{R_{n-2} \text{ with } \alpha = \sqrt{2}, \beta = 1} \sqrt{3}Q_{n-3} + \sum_{i=0}^{n-4} Q_i \\ &\times \xrightarrow{R_{n-3} \text{ with } \alpha = \sqrt{3}, \beta = 1} \sqrt{4}Q_{n-4} + \sum_{i=0}^{n-5} Q_i \\ &\vdots \\ &\times \xrightarrow{R_2 \text{ with } \alpha = \sqrt{n-2}, \beta = 1} \sqrt{n-1}Q_1 + Q_0 \\ &\times \xrightarrow{R_1 \text{ with } \alpha = \sqrt{n-1}, \beta = 1} \sqrt{n}Q_0, \end{aligned} \tag{30}$$

where each arrow represents an application of R_i as in (28) with the given values of α and β. Hence, we map B to $\sqrt{n}Q_0 = \sqrt{n}X_0$, i.e., a single-qubit Pauli σ_x, via $n - 1$ rotations generated by two-qubit Pauli matrices Y_iX_{i-1}.

Let R denote this entire sequence of rotations, i.e.,

$$R = \prod_{i=(n-1)}^{i=1} R_i. \tag{31}$$

In terms of R, the above result is

$$RBR^\dagger = \sqrt{n}X_0, \tag{32}$$

which implies that

$$B = \sqrt{n}R^\dagger X_0 R. \tag{33}$$

This is all that is needed in the QTDA use case. Since $R^\dagger X_0 R$ is unitary, it can be implemented as a quantum circuit, and to obtain B the constant of proportionality can be included during classical postprocessing.

Extensions. In certain use cases (such as propagators in differential equations), it may be desired to implement the time evolution of B. For these cases, in order to implement the time evolution generated by B for a time t, we can apply R, then implement the time evolution generated by $\sqrt{n}X_0$ for time t (analytically), and then invert R. This exponentiation is achieved analytically and therefore does not suffer from any Trotterization error:

$$e^{-\mathrm{i}Bt} = R^{\dagger}e^{-\mathrm{i}\sqrt{n}X_0 t}R, \tag{34}$$

where R and R_i are as defined in (28) and (31). The cost of implementing this evolution is independent of t since we can classically precompute $\sqrt{n}t \mod 2\pi$ and implement the rotation generated by X_0 through this angle.

Finally, in applications where we require the non-Hermitian boundary operator ∂_k (e.g., computational geometry), we can apply projection operators on either side of B to compute $\partial_k = P_{k-1}BP_k$; see [41] for details on how to construct these projectors efficiently on a quantum computer.

IV. DISCUSSION: CIRCUIT DEPTH AND SHOTS

The above unitary form of the boundary operator (33) has depth $O(n)$ since there are $2(n-1)$ rotations and one X_0. To be precise, the dependence on n is contained in R, which is a sequence of $n-1$ two-qubit rotations, so in total we require $2(n-1)$ two-qubit rotations. To finalize the $O(n)$ discussion, we must explain only that each Pauli rotation (involving two qubits) can be implemented at constant depth. For example, the standard way to evolve a single Pauli string is to change the basis of each of the qubits affected by a σ_x or σ_y into the σ_z basis (we have only two such qubits for each rotation, which is independent of n), followed by rotation around the z axis while accounting for parity. Therefore, the two-qubit rotations are independent of n, and the overall depth of (33) is $O(n)$.

With the boundary operator performed analytically there is significant savings in the number of shots needed compared to the approximate method of the original fermionic boundary operator (21) as used in Δ [41]. In order to account for the Trotterization error, which for an n-qubit Hamiltonian is written as $B = \sum_{i=0}^{n-1} Q_i$, we have the first-order approximation $e^{-i\sum_{i=0}^{n-1} Q_i t} = \prod_{i=0}^{n-1} e^{-iQ_i t} + O(t^2)$. Therefore, the error due Trotterization is $\epsilon_T \sim O(t^2)$. Then the Taylor-series expansion for the Hamiltonian simulation is given by $e^{-iBt} = 1 - iBt - B^2t^2/2 + O(B^3t^3) = 1 - iBt + O(nt^2)$ since $B^2 = nI$. Solving for B yields $B = (e^{-iBt} + i + \epsilon_{TS})/t$, where $\epsilon_{TS} \sim O(nt^2)$. Therefore, including ϵ_T,

$$B = \left(\prod_{i=0}^{n-1} e^{-iQ_i t} + i + \epsilon_{TS} + \epsilon_T\right)/t.$$

Next, taking the expectation produces shot noise, $\epsilon_{\text{shot}} \sim \frac{1}{\sqrt{N}}$. Similar to ϵ_T and ϵ_{TS}, ϵ_{shot} gets amplified by the fractional t denominator: ϵ_{shot}/t. Therefore, the overall error is

$$\epsilon = (\epsilon_T + \epsilon_{TS} + \epsilon_{\text{shot}})/t.$$

Now, each of these errors should be of the same order as the desired order of the precision ϵ. Handling each error independently and solving for t in the stronger Taylor-series constraint, we have $t \sim O(\epsilon/n)$. From the shot-noise constraint and solving for N, we have $N \sim O(1/(\epsilon^2 t^2))$. Finally, inserting the above t dependence on ϵ yields $N \sim O(n^2/\epsilon^4)$ for the approximate version of the original fermionic boundary operator [41]. In contrast, for the analytic circuit presented in this paper, since there is no Trotterization and Taylor expansion error, the number of samples needed is $N \sim O(\frac{1}{\epsilon^2})$, which is a quadratic savings (and more for higher moments).

V. CONCLUSION

In this paper we provided a short-depth, $O(n)$, quantum circuit for the exact, analytical implementation of the full Hermitian boundary operator on a gate-based quantum computer. This is a significant improvement over previous approximate implementations, resulting in at least quadratic savings in the number of shots. In order to achieve this we formally proved that the boundary operator can be written as a sum of fermionic creation and annihilation operators. This connection between algebraic geometry and fermionic operators opens a potentially rich vein for further exploration. The fermionic representation together with the convenient property of pairwise Pauli anticommutation permits implementation via a short circuit consisting of a cascade of two-qubit rotations. With such a short-depth circuit for the full boundary operator, the door is open for the search for quantum implementations of many algorithms that have the boundary operator as a core component, including in the fields of differential equations and machine learning.

ACKNOWLEDGMENTS

Y.-H.H. would like to thank UK STFC for Grant No. ST/J00037X/2. V.J. is supported by the South African Research Chairs Initiative of the Department of Science and Innovation and the National Research Foundation Grant 78554 and by the Simons Foundation Mathematics and Physical Sciences Targeted Grant, 509116. W.K. is supported by the National Science Foundation, Grant No. DGE-1842474.

APPENDIX: EXPLICIT EXAMPLES

In this Appendix, we illustrate the formulas in the main text with the explicit example of $n = 3$. There are three vertices here, which we will call $v_{0,1,2}$. We then use binary degree-lexicographic representation of the eight simplices constructible from these vertices as follows:

Dimension	Binary Representation	Simplex
	000	$\emptyset$
0	001	v_0
	010	v_1
	100	v_2
1	011	line(v_0, v_1)
	101	line(v_0, v_2)
	110	line(v_1, v_2)
2	111	triangle(v_0, v_1, v_2)

(A1)

Let us check a few of the boundary operators as defined in Sec. II A 1. Consider $\partial_1^{(3)}$, which should take a line to its boundary points, with appropriate ± 1 sign depending on the direction of the edge. Taking $|s_1\rangle = 011$, we have only two terms contributing since the Kronecker δ picks out v_1 and v_0, i.e., $i = 0, 1$,

$$\begin{aligned}\partial_1^{(3)}|s_1\rangle &= (-1)^{\sum_{j=1}^{2} s_1(j)}v_0 + (-1)^{\sum_{j=2}^{2} s_1(j)}v_1\\ &= (-1)^{1+0}v_0 + (-1)^0 v_1\\ &= v_1 - v_0. \qquad \text{(A2)}\end{aligned}$$

Next, we take $|s_2\rangle = 111$ and act upon it with $\partial_2^{(3)}$, whereupon all three terms in the sum contribute:

$$\begin{aligned}\partial_2^{(3)}|s_2\rangle &= (-1)^{\sum_{j=1}^{2} s_2(j)}v_0 + (-1)^{\sum_{j=2}^{2} s_2(j)}v_1\\ &\quad + (-1)^{\sum_{j=3}^{2} s_2(j)}v_2\\ &= (-1)^{1+1}v_0 + (-1)^1 v_1 + (-1)^0 v_2\\ &= v_0 - v_1 + v_2, \qquad \text{(A3)}\end{aligned}$$

which says that the boundaries of the triangle are the three edges with appropriate signs (± 1) depending on the directions of these edge.

1. The boundary operator $\partial^{(n)}$

We now write down the boundary operator $\partial^{(n)}$, constructed from the pieces $\partial_k^{(n)}$ given in (6), explicitly. The initial cases are simply

$$\partial^{(1)} = \begin{pmatrix}0 & 1\\ 0 & 0\end{pmatrix},$$

$$\partial^{(2)} = a_0 + a_1 = \sigma_z \otimes Q^+ + Q^+ \otimes I = \begin{pmatrix}0&1&0&0\\0&0&0&0\\0&0&0&-1\\0&0&0&0\end{pmatrix} + \begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&0&0\\0&0&0&0\end{pmatrix} = \begin{pmatrix}0&1&1&0\\0&0&0&1\\0&0&0&-1\\0&0&0&0\end{pmatrix},$$

$$\begin{aligned}\partial^{(3)} &= a_0 + a_1 + a_2 = \sigma_z \otimes \sigma_z \otimes Q^+ + \sigma_z \otimes Q^+ \otimes I + Q^+ \otimes I \otimes I\\
&= \begin{pmatrix}0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&-1&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&-1&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0\end{pmatrix} + \begin{pmatrix}0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&-1&0\\0&0&0&0&0&0&0&-1\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{pmatrix} + \begin{pmatrix}0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{pmatrix}\\
&= \begin{pmatrix}0&1&1&0&1&0&0&0\\0&0&0&1&0&1&0&0\\0&0&0&-1&0&0&1&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&-1&-1&0\\0&0&0&0&0&0&0&-1\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0\end{pmatrix}. \qquad \text{(A4)}\end{aligned}$$

a. Nilpotency

Let us check that the boundary operator is nilpotent, as is required from homology:

Lemma 2. The boundary operator satisfies $\partial^{(n)} \cdot \partial^{(n)} = 0$.

Proof. We can proceed by induction. Immediately, we check that $(\partial^{(1)})^2 = (\partial^{(2)})^2 = (\partial^{(3)})^2 = 0$, where 0 is the $2^n \times 2^n$ matrix of zeros; thus, the initial terms are fine. Next, we assume that $(\partial^{(n)})^2 = 0$ as the induction hypothesis. Now, we have (the third line uses the so-called mixed-product rule for Kronecker and dot products)

$$\begin{aligned}(\partial^{(n+1)})^2 &= \left(\partial^{(n)} \otimes I + \sigma_z^{\otimes n} \otimes Q^+\right)^2\\
&= \left(\partial^{(n)} \otimes I\right)^2 + \left(\sigma_z^{\otimes n} \otimes Q^+\right)^2 + (\partial^{(n)} \otimes I)\cdot(\sigma_z^{\otimes n} \otimes Q^+) + (\sigma_z^{\otimes n} \otimes Q^+)\cdot(\partial^{(n)} \otimes I)\\
&= (\partial^{(n)})^2 \otimes I^2 + \left(\sigma_z^{\otimes n}\right)^2 \otimes (Q^+)^2 + (\partial^{(n)} \cdot \sigma_z^{\otimes n}) \otimes (I \cdot Q^+)\\
&\quad + (\sigma_z^{\otimes n} \cdot \partial^{(n)}) \otimes (Q^+ \cdot I) \qquad [\text{since } (A \otimes B)\cdot(C \otimes D) = (AC) \otimes (BD), \text{ the mixed-product identity}]\\
&= 0 + 0 + (\sigma_z^{\otimes n} \cdot \partial^{(n)} + \partial^{(n)} \cdot \sigma_z^{\otimes n}) \otimes v \qquad [\text{since } (\partial^{(n)})^2 = (Q^+)^2 = 0]. \qquad \text{(A5)}\end{aligned}$$

Thus, it suffices to prove the anticommutation

$$\left\{\sigma_z^{\otimes n},\ \partial^{(n)}\right\} = \sigma_z^{\otimes n}\cdot\partial^{(n)} + \partial^{(n)}\cdot\sigma_z^{\otimes n} = 0. \tag{A6}$$

To see this, we set up another, separate induction. Clearly, the initial terms check out: $\{\partial^{(1)},\ \sigma_z\} = \{\partial^{(2)},\ \sigma_z^{\otimes 2}\} = 0$. Now, let the induction hypothesis be

$$\left\{\sigma_z^{\otimes n},\ \partial^{(n)}\right\} = 0, \tag{A7}$$

and consider

$$\begin{aligned}\left\{\sigma_z^{\otimes(n+1)},\ \partial^{(n)}\right\} &= \left\{\sigma_z^{\otimes(n+1)},\ \partial^{(n)}\otimes I + \sigma_z^{\otimes n}\otimes Q^+\right\}\\ &= \left\{\sigma_z^{\otimes n}\otimes\sigma_z,\ \partial^{(n)}\otimes I + \sigma_z^{\otimes n}\otimes Q^+\right\}\\ &= \left\{\sigma_z^{\otimes n},\ \partial^{(n)}\right\}\otimes(\sigma_z\cdot I) + (\sigma_z^{\otimes n}\cdot\sigma_z^{\otimes n})\\ &\quad\otimes\left\{\sigma_z,\ Q^+\right\}.\end{aligned} \tag{A8}$$

Now, the first term vanishes by the induction hypothesis, and the second term also vanishes since $\{\sigma_z,\ Q^+\} = \{\partial^{(1)},\ \sigma_z\} = 0$, the initial term. ■

b. Hamiltonian

We can also check the explicit form of the Hamiltonian $\exp(iB^{(n)}t) = \exp\{i[\partial^{(n)} + (\partial^{(n)})^\dagger]\}$ against (34).

$n = 1$.. Here, we have $B^{(1)} = \partial^{(1)} + (\partial^{(1)})^\dagger = \left(\begin{smallmatrix}0 & 1\\ 1 & 0\end{smallmatrix}\right)$, so that

$$\exp(iB^{(1)}t) = \begin{pmatrix}\cos(t) & -i\sin(t)\\ -i\sin(t) & \cos(t)\end{pmatrix}. \tag{A9}$$

On the right-hand side of the theorem, we have $R = R_0R_1$, where $R_0 = I_2$ and $R_1 = \exp[\frac{1}{2}Q_0Q_1\text{atan2}(0,1)] = I_2$. Hence, the right-hand side is just $\exp(-iX_0t) = \exp(-i\sigma_x t)$. Recalling that for integer n,

$$\sigma_x^{2n-1} = \sigma_x = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, \qquad \sigma_x^{2n} = I_2, \tag{A10}$$

we have equality.

$n = 2$.. From the forms of $\partial^{(n)}$ we find

$$\exp(iB^{(2)}t) = \begin{pmatrix}\cos(\sqrt{2}t) & -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & 0\\ -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & \cos(\sqrt{2}t) & 0 & -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}}\\ -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & 0 & \cos(\sqrt{2}t) & \frac{i\sin(\sqrt{2}t)}{\sqrt{2}}\\ 0 & -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & \frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & \cos(\sqrt{2}t)\end{pmatrix}. \tag{A11}$$

We can now cross-check this explicit matrix against Eq. (34). We have, from $\partial^{(2)}$, that $R = R_1 = \exp[\frac{1}{2}Q_0Q_1\text{atan2}(\sqrt{2}-1, 1)]$. Thus,

$$R = R_1 = \exp\left[\frac{\pi}{8}(a_0 + a_0^\dagger)(a_1 + a_1^\dagger)\right] = \begin{pmatrix}\cos\left(\frac{\pi}{8}\right) & 0 & 0 & \sin\left(\frac{\pi}{8}\right)\\ 0 & \cos\left(\frac{\pi}{8}\right) & \sin\left(\frac{\pi}{8}\right) & 0\\ 0 & -\sin\left(\frac{\pi}{8}\right) & \cos\left(\frac{\pi}{8}\right) & 0\\ -\sin\left(\frac{\pi}{8}\right) & 0 & 0 & \cos\left(\frac{\pi}{8}\right)\end{pmatrix}. \tag{A12}$$

Hence,

$$R^\dagger\exp(-i\sqrt{2}(a_0 + a_0^\dagger)t)R = \begin{pmatrix}\cos(\sqrt{2}t) & -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & 0\\ -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & \cos(\sqrt{2}t) & 0 & -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}}\\ -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & 0 & \cos(\sqrt{2}t) & \frac{i\sin(\sqrt{2}t)}{\sqrt{2}}\\ 0 & -\frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & \frac{i\sin(\sqrt{2}t)}{\sqrt{2}} & \cos(\sqrt{2}t)\end{pmatrix}, \tag{A13}$$

and we have perfect agreement of (A10) and (A12).

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

[2] M. Hirvensalo, *Quantum Computing*, Natural Computing Series, 2nd ed. (Springer, Berlin, Heidelberg, 2004).

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2010).

[4] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature (London) **549**, 195 (2017).

[5] M. Schuld, I. Sinayskiy, and F. Petruccione, An introduction to quantum machine learning, Contemp. Phys. **56**, 172 (2015).

[6] M. Schuld and N. Killoran, Quantum Machine Learning in Feature Hilbert Spaces, Phys. Rev. Lett. **122**, 040504 (2019).

[7] S. Lloyd, S. Garnerone, and P. Zanardi, Quantum algorithms for topological and geometric analysis of data, Nat. Commun. **7**, 10138 (2016).

[8] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning

with quantum-enhanced feature spaces, Nature (London) **567**, 209 (2019).

[9] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[10] A. Zhao, A. Tranter, W. M. Kirby, S. F. Ung, A. Miyake, and P. J. Love, Measurement reduction in variational quantum algorithms, Phys. Rev. A **101**, 062322 (2020).

[11] S. Aaronson, Read the fine print, Nat. Phys. **11**, 291 (2015).

[12] Y. Liu, S. Arunachalam, and K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, Nat. Phys. **17**, 1013 (2021).

[13] C. H. Edwards, D. E. Penney, and D. T. Calvis, *Differential Equations and Boundary Value Problems* (Pearson Education, 2016).

[14] D. W. Berry, High-order quantum algorithm for solving linear differential equations, J. Phys. A **47**, 105301 (2014).

[15] S. Lloyd, G. De Palma, C. Gokler, B. Kiani, Z.-W. Liu, M. Marvian, F. Tennie, and T. Palmer, Quantum algorithm for nonlinear differential equations, arXiv:2011.06571.

[16] O. C. Zienkiewicz, R. L. Taylor, P. Nithiarasu, and J. Z. Zhu, *The Finite Element Method*, (McGraw-Hill, London, 1977), Vol. 3.

[17] A. Montanaro and S. Pallister, Quantum algorithms and the finite element method, Phys. Rev. A **93**, 032324 (2016).

[18] F. Chung, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* (Rutgers, NJ, 1993), pp. 21–36.

[19] H. Mo, M. Bai, D. Lin, and J.-C. Roegiers, Study of flow and transport in fracture network using percolation theory, Appl. Math. Model. **22**, 277 (1998).

[20] A. Muhammad and M. Egerstedt, Control using higher order Laplacians in network topologies, *Proceedings of 17th International Symposium on Mathematical Theory of Networks and System* (Citeseer, 2006), pp. 1024–1038.

[21] E. Brisson, Representing geometric structures in d dimensions: Topology and order, Discrete Comput. Geom. **9**, 387 (1993).

[22] A. Zomorodian and G. Carlsson, Computing persistent homology, Discrete Comput. Geom. **33**, 249 (2005).

[23] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, Computer Science Series (McGraw-Hill, New York, 1995).

[24] M. K. Pietikainen, *Texture Analysis in Machine Vision*, Series In Machine Perception And Artificial Intelligence (World Scientific Publishing Company, 2000).

[25] H. O. Fattorini, Boundary control systems, SIAM J. Control **6**, 349 (1968).

[26] G. Golo, V. Talasila, A. Van Der Schaft, and B. Maschke, Hamiltonian discretization of boundary control systems, Automatica **40**, 757 (2004).

[27] A. Hatcher, *Algebraic Topology* (Cambridge University Press, 2002).

[28] J. R. Munkres, *Elements of Algebraic Topology* (CRC Press, Boca Raton, FL, 2018).

[29] A. Y. Ng, M. I. Jordan, and Y. Weiss, On spectral clustering: Analysis and an algorithm, *Advances in Neural Information Processing Systems*, edited by T. Dietterich, S. Becker, and Z. Ghahramani, Vol. 14 (MIT Press, 2002), pp. 849–856.

[30] J. Shi and J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. **22**, 888 (2000).

[31] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation, IEEE Trans. Knowl. Data Eng. **19**, 355 (2007).

[32] K. Kamvar, S. Sepandar, K. Klein, D. Dan, M. Manning, and C. Christopher, Spectral learning, *Proceedings of 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 2003* (Stanford InfoLab, Acapulco, Mexico, 2003).

[33] K. Zhang and J. T. Kwok, Clustered Nyström method for large scale manifold learning and dimension reduction, IEEE Trans. Neural Networks **21**, 1576 (2010).

[34] M. Welling and T. N. Kipf, Semi-supervised classification with graph convolutional networks, *Journal of International Conference on Learning Representations* (ICLR 2017).

[35] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, Graph neural networks: A review of methods and applications, AI Open **1**, 57 (2020).

[36] Z. Yan, T. Ma, L. Gao, Z. Tang, and C. Chen, *Proceedings of the 38th International Conference on Machine Learning*, edited by M. Meila and T. Zhang, Proceedings of Machine Learning Research, Vol. 139 (PLMR 2021), pp. 11659–11669.

[37] R. Ghrist, Barcodes: The persistent topology of data, Bull. Am. Math. Soc. **45**, 61 (2008).

[38] P. Bubenik, Statistical topological data analysis using persistence landscapes, J. Mach. Learn. Res. **16**, 77 (2015).

[39] L. Wasserman, Topological data analysis, Annu. Rev. Stat. Its Appl. **5**, 501 (2018).

[40] C. Gyurik, C. Cade, and V. Dunjko, Towards quantum advantage for topological data analysis, arXiv:2005.02607.

[41] S. Ubaru, I. Y. Akhalwaya, M. S. Squillante, K. L. Clarkson, and L. Horesh, Quantum topological data analysis with linear depth and exponential speedup, arXiv:2108.02811.

[42] C. Cade and P. M. Crichigno, Complexity of supersymmetric systems and the cohomology problem, arXiv:2107.00011.

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

[44] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, Simulation of electronic structure Hamiltonians using quantum computers, Mol. Phys. **109**, 735 (2011).

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

[46] A. F. Izmaylov, T.-C. Yen, R. A. Lang, and V. Verteletskyi, Unitary partitioning approach to the measurement problem in the variational quantum eigensolver method, J. Chem. Theory Comput. **16**, 190 (2020).

[47] M. Hochbruck and A. Ostermann, Exponential integrators, Acta Numer. **19**, 209 (2010).

[48] P. Jordan and E. Wigner, Über das paulische äquivalenzverbot, Z. Phys. **47**, 631 (1928).

[49] S. B. Bravyi and A. Yu. Kitaev, Fermionic quantum computation, Ann. Phys. (NY) **298**, 210 (2002).

[50] A. Ralli, P. J. Love, A. Tranter, and P. V. Coveney, Implementation of measurement reduction for the variational quantum eigensolver, Phys. Rev. Research **3**, 033195 (2021).