Towards Quantum Advantage via Topological Data Analysis

Casper Gyurik,[1] Chris Cade,[2] and Vedran Dunjko[1]
[1]*Leiden Institute of Advanced Computer Science, Universiteit Leiden, 2333 CA Leiden, Netherlands*
[2]*QuSoft, Centrum Wiskunde & Informatica, 1098 XG Amsterdam, Netherlands*
(Dated: December 18, 2020)

Even after decades of quantum computing development, examples of generally useful quantum algorithms with exponential speedups over classical counterparts are scarce. Recent progress in quantum algorithms for linear-algebra positioned quantum machine learning (QML) as a potential source of such useful exponential improvements. Yet, in an unexpected development, a recent series of "dequantization" results has equally rapidly removed the promise of exponential speedups for several QML algorithms. This raises the critical question whether exponential speedups of other linear-algebraic QML algorithms persist. In this paper, we study the quantum-algorithmic methods behind the algorithm for topological data analysis of Lloyd, Garnerone and Zanardi through this lens. We provide evidence that the problem solved by this algorithm is classically intractable by showing that its natural generalization is as hard as simulating the one clean qubit model – which is widely believed to require superpolynomial time on a classical computer – and is thus very likely immune to dequantizations. Based on this result, we provide a number of new quantum algorithms for problems such as rank estimation and complex network analysis, along with complexity-theoretic evidence for their classical intractability. Furthermore, we analyze the suitability of the proposed quantum algorithms for near-term implementations. Our results provide a number of useful applications for full-blown, and restricted quantum computers with a guaranteed exponential speedup over classical methods, recovering some of the potential for linear-algebraic QML to become a killer application of quantum computers.

I. INTRODUCTION

Quantum machine learning (QML) is a rapidly growing field [1, 2] that has brought forth numerous proposals regarding ways for quantum computers to help analyze data. Several of these proposals involve using quantum algorithms for linear algebra – most notably Harrow, Hassidim and Lloyd's matrix inversion algorithm [3] – to exponentially speed up tasks in machine learning. Other proposals such as the use of parameterized quantum circuits [4–6] provide a different approach based on identifying genuinely new quantum learning models (rather than speedups of established methods), which are more amenable to near-term quantum computing restrictions. These QML proposals have all been hailed as possible examples of quantum computing's "killer application": genuinely and broadly useful quantum algorithms which superpolynomially outperform their best known classical counterparts, which are very rare even if full-blown quantum computing is assumed.

However, previously speculated exponential speedups of linear-algebraic QML proposals were revealed to actually be at most polynomial speedups, as exponentially faster classical algorithms were devised that operate under analogous assumptions [7, 8]. Nevertheless, practically relevant polynomial speedups may persist [9, 10]. While even quadratic speedups have obvious appeal on paper, recent analysis involving concrete near-term device properties revealed that low-degree polynomial improvements are not expected to translate to real-world advantages due to various overheads [11]. Thus, finding superpolynomial speedups is of great importance, especially in the early days of practical quantum computing. Consequently, it is imperative to re-examine other linear-algebraic QML algorithms to ensure that speculated exponential quantum speedups will not be lost due to development of better classical algorithms.

In this paper, we focus on the quantum-algorithmic methods used by the comparatively less studied algorithm for topological data analysis (TDA) of Lloyd, Garnerone and Zanardi (LGZ) [12], and on the TDA problem itself. We show that the underlying linear-algebraic methods are "safe" against general dequantization approaches of the type introduced in [7, 8], and that the corresponding computational problem is generally classically intractable, under widely-believed complexity-theoretic assumptions. This further establishes the potential of these methods to be a source of useful quantum algorithms with exponential speedups over classical methods, which we demonstrate by connecting them to practical problems in machine learning and complex network analysis. Additionally, we discuss the possibilities of near-term implementations of these quantum methods, which helps position TDA and related problems in the domain of NISQ [13] devices as well. The main contributions of this paper are as follows:

- We provide evidence that TDA (as solved by the LGZ algorithm) is classically intractable. Specifically, we show that a generalization of the TDA problem is as hard as simulating the one clean qubit model of quantum computation, which is widely believed to require exponential time on a classical computer.
- We provide efficient quantum algorithms for rank estimation and complex network analysis based on the

quantum algorithmic methods underlying the LGZ algorithm, along with complexity-theoretic evidence for the classical hardness of the underlying problems.

- We analyze the possibilities and challenges of near-term implementations of the quantum-algorithmic methods of the LGZ algorithm, focusing on providing several techniques to reduce the required resources, making it more suitable for low-qubit computations.

We note that while our results do not imply that the narrow TDA problem as solved by the algorithm of LGZ is itself classically intractable (our generalization, however, is shown to be classically intractable), they do eliminate the possibility of a generic dequantization method that does not take into account the specifics of the TDA problem. This is also the case for our extension to complex network analysis, whereas our results show that our extension to rank estimation is itself entirely classically intractable.

The paper is organized as follows. For didactic purposes we provide in Section II a detailed description of the quantum algorithm of LGZ and the background on topological data analysis. Our main results on the underlying classical hardness are presented in Section III. In Section IV we discuss how to extend the applicability of the methods used by the quantum algorithm of LGZ, and we discuss the potential for near-term implementations in Section V. We finish the paper with a discussion of our results in Section VI.

II. TOPOLOGICAL DATA ANALYSIS AND THE QUANTUM ALGORITHM

Topological data analysis is a recent approach to data analysis that extracts robust features from a dataset by inferring properties of the shape of the data. This is perhaps best explained in analogy to a better-known method: much like how principal component analysis extracts features (i.e., the singular values characterizing the spread of the data in the directions of highest variance) that are invariant under translation and rotation of the data, topological data analysis goes a step further and extract features that are also invariant under bending and stretching of the data by inferring properties of its general shape. Because of this invariance of the extracted features, topological data analysis techniques are inherently robust to noise in the data.

The theory behind topological data analysis is fairly extensive, but most of it we will not need for our purpose. Namely, we can set most of the topology aside and tackle the issue in linear-algebraic terms, which are well-suited for quantum approaches. In this section we introduce the relevant linear-algebraic concepts, and we briefly review the quantum algorithm for topological data analysis of Lloyd, Garnerone and Zanardi (LGZ) [12].

A. Background and definitions

In topological data analysis the dataset is typically a point cloud (i.e., a collection of points in some ambient space) and the aim is to extract the shape of the underlying data (i.e., the 'source' of these points). This is done by constructing a connected object – called a *simplicial complex* – composed of points, lines, triangles and their higher-dimensional counterparts, whose shape one can study. After constructing the simplicial complex, features of the shape of the data – in particular, the number of connected components, holes, voids and higher-dimensional counterparts – can be extracted using linear-algebraic computations based on *homology*. An overview of this procedure can be found in Figure 2.

Consider a dataset of points $\{x_i\}_{i=1}^n$ embedded in some space equipped with a distance function d (typically $\mathbb{R}^m$ equipped with the Euclidean distance). The construction of the simplicial complex from this point cloud proceeds as follows. First, one constructs a graph by connecting datapoints that are "close" to each other. This is done by choosing a *grouping scale* ϵ (defining which points are considered "close") and connecting all datapoints that are within ϵ distance from each other. This yields the graph $G = ([n], E_\epsilon)$, with vertices $[n] := \{1, \ldots, n\}$ and edges

$$E_\epsilon = \{(i,j) \mid d(x_i, x_j) \leq \epsilon\}.$$

After having constructed this graph, one relates to it a particular kind of simplicial complex called a *clique complex*, by associating its cliques (i.e., complete subgraphs) with the building blocks of a simplicial complex[1]. That is, a 2-clique is considered a line, a 3-clique a triangle, a 4-clique a tetrahedron, and $(k+1)$-cliques the k-dimensional counterparts[2].

To fix the notation, let $\mathrm{Cl}_k(G) \subset \{0,1\}^n$ denote the set of $(k+1)$-cliques in G – where we encode a subset $\{i_1, \ldots, i_k\} \subset [n]$ as an n-bit string where the indices i_k specify the positions of the ones in the bitstring – and let $\chi_k := |\mathrm{Cl}_k(G)|$ denote the number of these cliques. Throughout this paper, we will discuss everything in terms of clique complexes, as this is sufficient for our purposes and allows us to use the more familiar terminology of graph theory.

The constructed clique complex exhibits the features that we want to extract from our dataset – i.e., the number of k-dimensional holes. For example, in Figure 1 we see a clique complex where we can count three 1-dimensional holes. Interestingly, counting these holes can be done more elegantly using linear algebra by employing constructions from homology.

[1] The resulting simplicial complex coincides with the Vietoris-Rips complex common in topological data analysis literature [14].

[2] The shift in the indexing is due to different terminologies in graph theory and topology (e.g., in graph theory a triangle is called a 3-clique, whereas in topology it is called a 2-simplex).

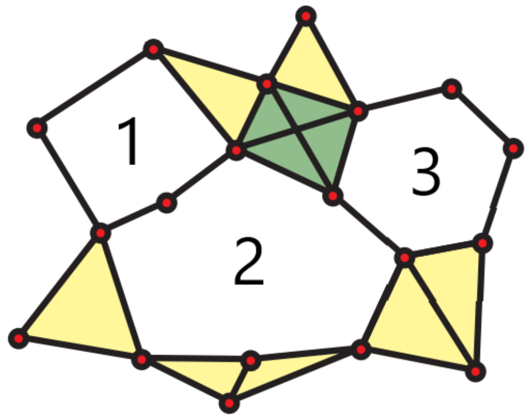


FIG. 1: Example of a clique complex with three 1-dimensional holes (adapted from [14]).

To extract these features using linear algebra, embed the clique complex into a Hilbert space $\mathcal{H}_k^G$, by raising the set of bitstrings that specify $(k+1)$-cliques to labels of orthonormal basis vectors. Note that $\mathcal{H}_k^G$ is a subspace of the Hilbert space $\mathcal{H}_k$ spanned by computational basis states with Hamming weight[3] $k+1$. Moreover, each $\mathcal{H}_k$ is an $\binom{n}{k+1}$-dimensional subspace of the entire n-qubit Hilbert space $\mathbb{C}^{2^n}$, and $\mathbb{C}^{2^n} \simeq \bigoplus_{k=-1}^{n-1} \mathcal{H}_k$.

The next step towards extracting features using linear algebra involves studying properties of the *boundary maps* $\partial_k : \mathcal{H}_k \to \mathcal{H}_{k-1}$, which are defined by linearly extending the action on the basis states given by

$$\partial_k \left|j\right\rangle \coloneqq \sum_{i=0}^{k} (-1)^i \left|\widehat{j(i)}\right\rangle, \tag{1}$$

where $\widehat{j(i)}$ denotes the n-bit string of Hamming weight k that encodes the subset obtained by removing the i-th element from the subset encoded by j (i.e., we set the i-th one in the bitstring j to zero). By considering the restriction of ∂_k to $\mathcal{H}_k^G$ – which we denote by ∂_k^G – these boundary maps can encode the connectivities of the graph G, in which case their image and kernel encode various properties of the corresponding clique complex. Intuitively, these boundary maps map a $(k+1)$-clique to a superposition (i.e., a linear combination) of all k-cliques that it contains, as seen in Eq. 1.

These boundary maps allow one to extract features of the shape of a clique complex by studying their images and kernels, and in particular their quotients. Specifically, the quotient space

$$H_k(G) \coloneqq \ker \partial_k^G / \operatorname{Im} \partial_{k+1}^G, \tag{2}$$

which is called the k-th *homology group*, captures features of the shape of the underlying clique complex. The main feature is the k-th *Betti number* β_k^G, which is defined as the dimension of the k-th homology group, i.e.,

$$\beta_k^G \coloneqq \dim H_k(G).$$

By construction, the k-th Betti number is equal to the number of k-dimensional holes in the clique complex.

The main problem in topological data analysis that we study in this paper is the computation of Betti numbers. To do so, we study the *combinatorial Laplacians* [15], which are defined as

$$\Delta_k^G = \left(\partial_k^G\right)^\dagger \partial_k^G + \partial_{k+1}^G \left(\partial_{k+1}^G\right)^\dagger.$$

These combinatorial Laplacians can be viewed as generalized (or rather, higher-order) graph Laplacians in that they encode the connectivity between cliques in the graph as opposed to encoding the connectivity between individual vertices. We study the combinatorial Laplacians because the discrete version of the Hodge theorem [15] tells us that

$$\dim \ker \left(\Delta_k^G\right) = \beta_k^G, \tag{3}$$

which is often used as a more convenient way to compute Betti numbers [16], particularly in the case of the quantum algorithm that we discuss in the next section.

In conclusion, if the clique complex is constructed from a point cloud according to the construction discussed above, then computing these Betti numbers can be viewed as a method to extract features of the shape of the data (specifically, the number of holes are present at scale ϵ). By recording Betti numbers across varying scales ϵ in a so-called *barcode* [14], one can discern which holes are “real” and which are “noise”, resulting in feature extraction that is robust to noise in the data.

B. Quantum algorithm for Betti number estimation

The algorithm for Betti number estimation of Lloyd, Garnerone and Zanardi (LGZ) [12] utilizes Hamiltonian simulation and phase estimation to estimate the dimension of the kernel (i.e., the *nullity*) of the combinatorial Laplacian (which by Eq. 3 is equal to the corresponding Betti number). To make our presentation self-contained, we review this quantum algorithm for Betti number estimation (for a more in-depth review see [18]).

Estimating the nullity of a sparse Hermitian matrix can be achieved using some of the most fundamental quantum-algorithmic primitives. Namely, using Hamiltonian simulation and quantum phase estimation one can estimate the eigenvalues of the Hermitian matrix, given that the eigenvector register starts out in an eigenstate. Moreover, if instead the eigenvector register starts out in the maximally mixed state (which can be thought of as

[3] The Hamming weight of a bitstring is the number of 1s in it.

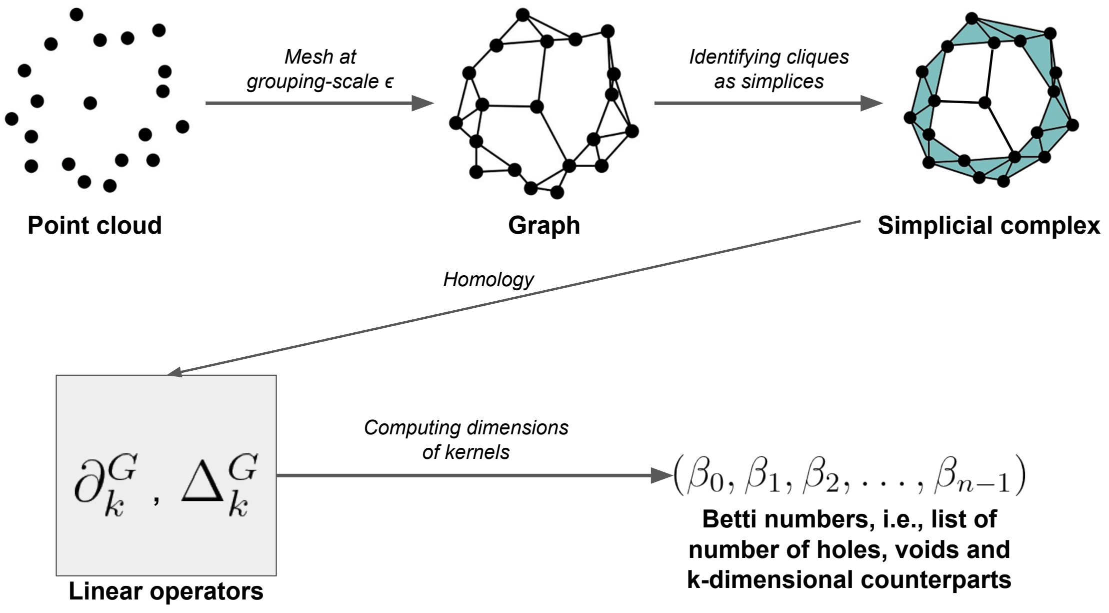

Point cloud
Mesh at grouping-scale ϵ
Graph
Identifying cliques as simplices
Simplicial complex
Homology
∂_k^G , Δ_k^G
Linear operators
Computing dimensions of kernels
$(\beta_0, \beta_1, \beta_2, \ldots, \beta_{n-1})$
Betti numbers, i.e., list of number of holes, voids and k-dimensional counterparts

FIG. 2: The pipeline of topological data analysis (adapted from [17]).

a random choice of an eigenstate), then measurements of the eigenvalue register produce approximations of eigenvalues, sampled uniformly at random from the set of all eigenvalues. This routine is then repeated to estimate the nullity by simply computing the frequency of zero eigenvalues (recall that the dimension of the kernel is equal to the multiplicity of the zero eigenvalue). Note that this procedure does not strictly speaking estimate the nullity, but rather the number of small eigenvalues, where the threshold is determined by the precision of the quantum phase estimation (see Section II B 1 for more details).

In summary, the steps of the quantum algorithm for Betti number estimation of LGZ are as follows:

Quantum algorithm for Betti number estimation [12]

1. For $i = 1, \ldots, M$ repeat:
   (a) Prepare the state:
   $$\rho_k^G = \frac{1}{|\dim \mathcal{H}_k^G|} \sum_{j \in \mathrm{Cl}_k(G)} |j\rangle\langle j| . \tag{4}$$
   (b) Apply quantum phase estimation to the unitary e^{iB}, with the eigenvector register starting out in the state ρ_k^G.
   (c) Measure the eigenvalue register to obtain an approximation $\widetilde{\lambda}_i$.
2. Output the frequency of zero eigenvalues:
   $$\left|\{i \mid \widetilde{\lambda}_i = 0\}\right| / M.$$

In Step 1(a), Grover's algorithm is used to prepare the uniform superposition over $\mathcal{H}_k^G$, from which one can prepare the state ρ_k^G by applying a CNOT gate to each qubit of the uniform superposition into some ancilla qubits and tracing those out. When given access to the adjacency matrix of G, one can check in $\mathcal{O}\left(k^2\right)$ operations whether a bitstring $j \in \{0,1\}^n$ encodes a valid k-clique and mark them accordingly in the application of Grover's algorithm. By cleverly encoding Hamming weight k strings we can avoid searching over all n-bit strings, which requires $\mathcal{O}(nk)$ additional gates per round of Grover's algorithm [18]. Hence, the runtime of this first step is

$$\mathcal{O}\left(k^2\sqrt{\binom{n}{k+1}/\chi_k}\right) = \mathcal{O}\left(k^2\sqrt{n^{k+1}/\chi_k}\right),$$

where χ_k denotes the number of $(k+1)$-cliques. This runtime is polynomial in the number of vertices n when

$$\binom{n}{k+1}/\chi_k \in \mathcal{O}(\mathrm{poly}(n)). \tag{5}$$

Throughout this paper we say that a graph is *clique-dense* if it satisfies Eq. 5. Note that ρ_k^G can of course also be directly prepared without the use of Grover's algorithm by using rejection sampling: choose a subset uniformly at random and accept it if it encodes a valid clique. This is quadratically less efficient, however it has advantages if one has near-term implementations in mind, as it is a completely classical subroutine.

In Step 1(b), standard methods for Hamiltonian simulation of sparse Hermitian matrices are used together with quantum phase estimation to produce approximations of the eigenvalues of the simulated matrix. The matrix that LGZ simulate in this step is the *Dirac operator*, which is defined as

$$B_G = \begin{pmatrix} 0 & \partial_1^G & 0 & \dots & \dots & 0 \\ \left(\partial_1^G\right)^\dagger & 0 & \partial_2^G & \dots & \dots & 0 \\ 0 & \left(\partial_2^G\right)^\dagger & 0 & \ddots & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 & \partial_{n-1}^G \\ 0 & 0 & 0 & \dots & \left(\partial_{n-1}^G\right)^\dagger & 0 \end{pmatrix}$$

and satisfies

$$B_G^2 = \begin{pmatrix} \Delta_0^G & 0 & \dots & 0 \\ 0 & \Delta_2^G & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & \Delta_{n-1}^G \end{pmatrix}. \tag{6}$$

From Eq. 6 we gather that the probability of obtaining an approximation of an eigenvalue that is equal to zero is proportional to the nullity of the combinatorial Laplacian. Because B_G is an n-sparse Hermitian matrix with entries $0, -1$ and 1, to which we can implement sparse access using $\mathcal{O}(n)$ gates, we can implement e^{iB} using $\widetilde{\mathcal{O}}\left(n^2\right)$[‡] gates [19].

We remark that it is also possible to simulate Δ_k^G directly. Namely, as Δ_k^G is an n-sparse Hermitian matrix whose entries are bounded above by n, to which we can implement sparse access using $\mathcal{O}\left(n^3\right)$ gates, we can implement $e^{i\Delta_k^G}$ using $\widetilde{\mathcal{O}}\left(n^4\right)$ gates [19]. The disadvantage to directly simulating Δ_k^G is that it requires more gates. However, the advantage is that it can be done using fewer qubits, namely, $\log\binom{n}{k+1}$ qubits instead of n. Moreover, when the graph is clique-dense one can bypass Step 1(a) by padding Δ_k^G with all-zero rows and columns and letting the eigenvector start out in the maximally mixed state $I/2^n$ (see Section III A 1 for more details).

Let λ_{max} denote the largest eigenvalue and let λ_{min} denote the smallest nonzero eigenvalue of Δ_k^G. By scaling down the matrix one chooses to simulate (i.e., either B or Δ_k^G) by $1/\lambda_{max}$ to avoid multiples of 2π, we can tell whether an eigenvalue is equal to zero or not if the precision of the quantum phase estimation is at least $\lambda_{max}/\lambda_{min}$. By the Gershgorin circle theorem (which states that λ_{max} is bounded above by the maximum sum of absolute values of the entries of a column or row) we know that $\lambda_{max} \in \mathcal{O}(n)$. For the general case not much is known in terms of lower bounds on λ_{min} (see Section II B 1 for more detail). By taking into account the cost of the quantum phase estimation [20], the total runtime of Step 1(b) becomes $\widetilde{\mathcal{O}}\left(n^3/\lambda_{min}\right)$.

Finally, estimating $\beta_k^G/\dim\mathcal{H}_k^G$ up to additive precision ϵ can be done using $M \in \mathcal{O}\left(\epsilon^{-2}\right)$ repetitions of Step 1(a) through 1(c). This brings the total cost of estimating $\beta_k^G/\dim\mathcal{H}_k^G$ up to additive precision ϵ to

$$\widetilde{\mathcal{O}}\left(\left(k^2\sqrt{n^{k+1}/\chi_k} + n^3/\lambda_{min}\right)/\epsilon^2\right).$$

In conclusion, the quantum algorithm for Betti number estimation runs in time polynomial in n under two conditions. First, the graph has to be clique-dense, i.e., it has to satisfy Eq. 5 (more generally, we need an efficient clique-sampling routine, as discussed in Section IV A). Secondly, the smallest nonzero eigenvalue of the combinatorial Laplacian has to be at least inverse polynomial in n. If both these conditions are satisfied, then the quantum algorithm for Betti number estimation achieves an exponential speedup over the best known classical algorithms if the size of the combinatorial Laplacian – i.e., the number of $(k+1)$-cliques – scales exponentially in n (see Section II B 2 for more details).

1. *Approximate Betti numbers*

As mentioned in the previous section, the quantum algorithm for Betti number estimation does not strictly speaking estimate the Betti number (i.e., the nullity of the combinatorial Laplacian), but rather the number of small eigenvalues of the combinatorial Laplacian. This is because little is known in terms of lower bounds for the smallest nonzero eigenvalue of combinatorial Laplacians, and hence it is unclear to what precision one has to estimate the eigenvalues in the quantum phase estimation. In any case, it is conjectured that for high-dimensional simplicial complexes the smallest nonzero eigenvalue is at least inverse polynomial in n [16], which would imply that quantum phase estimation can in time $\mathcal{O}(\mathrm{poly}(n))$ determine whether an eigenvalue is exactly equal to zero.

Even without knowing a lower bound on the smallest nonzero eigenvalue of the combinatorial Laplacian, we can still perform quantum phase estimation up to some fixed inverse polynomial precision. The quantum algorithm for Betti number estimation then outputs an estimate of the number of eigenvalues of the combinatorial Laplacian that are smaller than this precision. Throughout this paper we will refer to this as *approximate Betti numbers*, which turn out to still convey information about the underlying graph (see Section IV C for more details).

2. *Comparison with classical algorithms*

The best known classical algorithms for computing Betti numbers run in time polynomial in the size of the combinatorial Laplacian, i.e., they run in time

$$\mathcal{O}\left(\mathrm{poly}\left(\chi_k\right)\right) \in \mathcal{O}\left(\mathrm{poly}\left(n^{k+1}\right)\right).$$

These classical algorithms use either a variant of the power method [16], or direct Gaussian elimination. We

[‡] $\widetilde{\mathcal{O}}()$ suppresses logarithmically growing factors.

do not know of any classical algorithm that focuses on estimating Betti numbers instead of exactly computing them. For approximate Betti number estimation (i.e., estimating the number of eigenvalues below a threshold), the best known classical algorithms run in time linear in the number of nonzero entries of the combinatorial Laplacian [21–23]. Since the combinatorial Laplacians are n-sparse, the number of nonzero entries of the combinatorial Laplacian – and hence also the runtime of the best known classical algorithm for approximate Betti number estimation – scales as

$$\mathcal{O}\left(n \cdot \chi_k\right) \in \mathcal{O}\left(n^{k+2}\right).$$

Recall that the quantum algorithm for Betti number estimation runs in time polynomial in n if the graph is clique-dense (i.e., it satisfies Eq. 5) and the smallest nonzero eigenvalue of the combinatorial Laplacian is at least inverse polynomial in n. For graphs that satisfy these conditions, we conclude that the quantum algorithm for Betti number estimation achieves an exponential speedup over the best known classical algorithms if the size of the combinatorial Laplacian – i.e., the number of $(k+1)$-cliques – scales exponential in n (which requires k to scale with n). The same conditions apply for exponential speedups for approximate Betti number estimation, except that we are not concerned with the size of the smallest nonzero eigenvalue of the combinatorial Laplacian.

In the next section, we will study the complexity of the problems related to the quantum-algorithmic methods used by the algorithm for Betti number estimation to see whether quantum computers can achieve a provable exponential advantage over classical computers. To be precise, we will study the complexity of generating approximations of eigenvalues that are sampled uniformly at random, that of estimating the number of eigenvalues below a given threshold, and that of estimating Betti numbers and approximate Betti numbers.

III. TOWARDS CLASSICAL HARDNESS OF BETTI NUMBER ESTIMATION

To show that quantum computers have an advantage over classical computers in topological data analysis, we would like to show that Betti number estimation requires exponential time on a classical computer. However, as discussed Section II B, the quantum algorithm for Betti number estimation does not precisely solve this problem, and is only efficient in certain specific regimes. In this section we study the classical hardness of the problem efficiently solved by the quantum algorithm for Betti number estimation. In particular, we show that a natural generalization of this problem is likely classically intractable.

We begin this section by formally defining the computational problems that capture the key steps in the quantum algorithm for Betti number estimation, and the problems related to topological data analysis that they solve. In particular, it is clear that the techniques used in the quantum algorithm for Betti number estimation can be used to estimate the number of small eigenvalues of arbitrary sparse Hermitian matrix, not just of combinatorial Laplacians. We take this as the starting point to define our natural generalization, which we call *low-lying spectral density* estimation.

Above all, we show that the generalization where we estimate the number of small eigenvalues not just for combinatorial Laplacians, but for a more general class of matrices is DQC1-hard. This suggests that the quantum-algorithmic methods behind the quantum algorithm for Betti number estimation may be a source of exponential separation between quantum and classical computers. In Section IV, we will demonstrate how the quantum-algorithmic methods captured by our generalization can be applied beyond Betti number estimation, suggesting even further that our separation can actually lead to useful computational advantage for quantum computers. We end this section with a discussion of the implications of our result and the open questions regarding the classical hardness of the variant of Betti number estimation that the quantum algorithm efficiently solves.

A. Problem definitions

In this section we formally define the computational problems whose hardness we will study. We begin by defining the problems that capture the key steps of the quantum algorithm for Betti number estimation. Afterwards, we define the problems related to topological data analysis that the quantum algorithm for Betti number estimation aims to solve. We end this section by discussing the precise relationships between these problems.

The input matrices that we consider are sparse positive semidefinite matrices. We call a $2^n \times 2^n$ positive semidefinite matrix *sparse* if at most $\mathcal{O}\left(\mathrm{poly}(n)\right)$ entries in each row are nonzero. A special class of sparse positive semidefinite matrices that we consider is the class of *log-local* Hamiltonians, i.e., n-qubit Hamiltonians that can be written as a sum

$$H = \sum_{j=1}^{m} H_j, \tag{7}$$

where each H_j acts on at most $\mathcal{O}\left(\log n\right)$ qubits and we assume that $m \in \mathcal{O}\left(\mathrm{poly}(n)\right)$.

Our problems take as input a specification of a sparse positive semidefinite matrix, and we consider the following two standard cases. First, we consider the case where the input matrix is specified in terms of *sparse access*. That is, the input matrix $H \in \mathbb{C}^{2^n \times 2^n}$ is specified by quantum circuits that let us to query the values of its entries and the locations of the nonzero entries. More precisely, we assume that we are given classical descriptions of $\mathcal{O}\left(\mathrm{poly}(n)\right)$-sized quantum circuits that implement the oracles O_H and $O_{H,loc}$, which map

$$O_H : |i,j\rangle |0\rangle \mapsto |i,j\rangle |H_{i,j}\rangle ,$$
$$O_{H,loc} : |j,\ell\rangle |0\rangle \mapsto |j,\ell\rangle |\nu(j,\ell)\rangle ,$$

where $0 \leq i,j,\ell \leq 2^n - 1$, and $\nu(j,\ell) \in \{0,\dots,2^n-1\}$ denotes the location of the ℓ-th nonzero entry of the j-th column of H. Secondly, for log-local Hamiltonians, we also consider specifying the input matrix H by its *local-terms* $\{H_j\}$ as in Eq. 7.

In order to define the problem of generating approximations of eigenvalues that are sampled uniformly at random, we fix a suitable notion of an approximation of a probability distribution. In particular, this notion needs to take into account that the algorithm may err on both the estimation of the eigenvalue, and on the probability with which it provides such an estimation. For this we use the following definition presented in [24]. Let p be some probability distribution over the eigenvalues of a positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$. That is, sampling according to p will output an eigenvalue λ_k with probability $p(\lambda_k)$, and $\sum_{k=0}^{2^n-1} p(\lambda_k) = 1$. In this context, a probability distribution q with finite support $Y_q \subset \mathbb{R}$ is said to be an (δ,μ)-*approximation* of p if it satisfies

$$\sum_{y\in Y_q\,:\,|y-\lambda_k|<\delta} q(y) \geq (1-\mu)p(\lambda_k), \;\; \forall k \in \{0,\dots,2^{n-1}\}.$$

Intuitively, this means that if we draw a sample according to q, then this sample will be δ-close to an eigenvalue λ_k with probability at least $(1-\mu)p(\lambda_k)$[1]. Using this definition, we define the problem of generating approximations of eigenvalues that are sampled uniformly at random from the set of all eigenvalues as follows.

Sparse uniform eigenvalue sampling (SUES)[2]
Input:
1) A sparse positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$, with $||H|| \leq \text{poly}(n)$.
2) An estimation precision $\delta \in \Omega\left(1/\text{poly}(n)\right)$.
3) An error probability $\mu \in \Omega\left(1/\text{poly}(n)\right)$.

Output: A sample drawn according to a (δ,μ)-approximation of the uniform distribution over the eigenvalues of H.

In the quantum algorithm for Betti number estimation, samples from SUES are used to estimate the number of eigenvalues of the combinatorial Laplacian that are close to zero. Clearly, this same idea can be used to estimate the number of eigenvalues that lie in some given interval for arbitrary sparse positive semidefinite matrices. This is called the *eigenvalue count* [25], which for a positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$ and eigenvalue thresholds $a,b \in \mathbb{R}_{\geq 0}$ is given by

$$N_H(a,b) = \frac{1}{2^n} \sum_{k\,:\,a\leq\lambda_k\leq b} 1,$$

where $\lambda_0 \leq \cdots \leq \lambda_{2^n-1}$ denote the eigenvalues of H. For a threshold $b \in \Omega\left(1/\text{poly}(n)\right)$, we shall refer to the quantity $N_H(0,b)$ as a *low-lying spectral density*. This precisely captures our notion of the number of eigenvalues close to zero as discussed before. We define the problem of estimating the low-lying spectral density as follows.

Low-lying spectral density estimation (LLSD)[3]
Input:
1) A sparse positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$, with $||H|| \leq \text{poly}(n)$.
2) A threshold $b \in \Omega\left(1/\text{poly}(n)\right)$.
3) Precision parameters $\delta, \epsilon \in \Omega\left(1/\text{poly}(n)\right)$.
4) A success probability $\mu > 1/2$.

Output: An estimate $\chi \in [0,1]$ that, with probability at least μ, satisfies

$$N_H\left(0,b\right) - \epsilon \leq \chi \leq N_H(0,b+\delta) + \epsilon.$$

To provide some intuition behind this definition, note that it is supposed to precisely capture the problem that is solved by repeatedly sampling from SUES and computing the frequency of the eigenvalues that lie below the given threshold. We therefore require the precision parameter δ due to the imprecisions in the quantum phase estimation algorithm. Moreover, the precision parameter ϵ is necessary due to the sampling error we incur by estimating a probability using its relative frequency.

Now that we have formally defined the problems that capture the key steps of the quantum algorithm for Betti number estimation, we define the problems related to topological data analysis that they allow us to solve. For these problems we consider the adjacency matrix of the graph to be the input, as this is the input to the quantum algorithm for Betti number estimation. We define the problem of estimating Betti numbers as follows.

Betti number estimation (BNE)[4]
Input:
1) The adjacency matrix of a graph $G = ([n], E)$.
2) An integer $0 \leq k \leq n-1$.
3) A precision parameter $\epsilon \in \Omega\left(1/\text{poly}(n)\right)$.
4) A success probability $\mu > 1/2$.

Output: An estimate $\chi \in [0,1]$ that, with probability at least μ, satisfies

$$\left|\chi - \frac{\beta_k^G}{\dim \mathcal{H}_k^G}\right| \leq \epsilon.$$

[1] This definition captures the distribution generated by quantum phase estimation: the eigenvector is chosen according to the distribution p specified by the input state, and the output is δ-close to the corresponding eigenvalue with probability at least $(1-\mu)$.

[2] In view of noisy quantum computers, it is interesting to consider distributions that are close to these (δ,μ)-approximation in total variation distance. Sampling such distributions can be less demanding, however, the precise hardness remains to be analyzed.

[3] The exact version of this problem is closely related to #P [26].

As discussed in Section II B 1, the quantum algorithm for Betti number estimation does not precisely solve the above problem. Namely, due to the lack of knowledge regarding lower bounds on the smallest nonzero eigenvalue of the combinatorial Laplacian, we are not always able to estimate the number of eigenvalues that are exactly equal to zero. Nonetheless, the quantum algorithm for Betti number estimation is still able to estimate the number of eigenvalues of the combinatorial Laplacian that are close to zero, which we called approximate Betti numbers. We define the problem of estimating approximate Betti numbers as follows.

Approximate Betti number estimation (ABNE)
Input:
1) The adjacency matrix of a graph $G = ([n], E)$.
2) An integer $0 \leq k \leq n-1$.
3) Precision parameters $\delta, \epsilon \in \Omega\left(1/\mathrm{poly}(n)\right)$.
4) A success probability $\mu > 1/2$.

Output: An estimate $\chi \in [0,1]$ that, with probability at least μ, satisfies

$$\frac{\beta_k^G}{\dim \mathcal{H}_k^G} - \epsilon \leq \chi \leq D_{\Delta_k^G}(0,\delta) + \epsilon.$$

We are now all set to precisely outline the problem that the quantum algorithm for Betti number estimation can efficiently solve. As discussed in Section II B 2, the quantum algorithm for Betti number estimation can efficiently solve ABNE, but only in certain regimes. In particular, one has to be able to efficiently prepare the maximally mixed state over $\mathcal{H}_k^G$ from the adjacency matrix of the graph. A sufficient condition for this is that the input graph is clique-dense (i.e., it should satisfy Eq. 5), as this allows us to efficiently prepare this maximally mixed state using either rejection sampling or Grover's algorithm. Thus, the problem that the quantum algorithm for Betti number estimation can efficiently solve is a restriction of ABNE, where one is promised that the input graph is clique-dense.

Next, we will study the complexity of LLSD as it is a generalization of the problem that the quantum algorithm for Betti number estimation efficiently solves. Namely, as we will show in the following section, we can use LLSD to directly solve the problem that the quantum algorithm for Betti number estimation efficiently solves. Note that the input to the quantum algorithm for Betti number estimation is the adjacency matrix, and not the combinatorial Laplacian. Therefore, in order to use LLSD to solve the problem that the quantum algorithm for Betti number estimation efficiently solves, one first has to construct the appropriate input to LLSD. As it is computationally too expensive to enumerate all cliques in your graph, we cannot take the straightforward approach of first computing the combinatorial Laplacian to construct the desired input to LLSD. Fortunately, we can still use LLSD to efficiently solve the problem that the quantum algorithm for Betti number estimation efficiently solves by simulating sparse access to a matrix that is obtained by padding the combinatorial Laplacian with all-zeros columns and rows (see Section III A 1 for more details). Note that in the quantum algorithm for Betti number estimation this is achieved by first preparing the maximally mixed state over $\mathcal{H}_k^G$.

1. *Relationships between the problems*

In the previous section we have formally defined the computational problems whose complexity we will study. In this section we examine reductions between LLSD and the problems related to topological data analysis in order to elucidate the precise relationships. An overview of the reductions can be found in Figure 3.

First, we discuss the relationship between LLSD and ABNE. It is clear that LLSD with a combinatorial Laplacian as input produces a solution to the corresponding instance of ABNE. It is clear that LLSD can solve ABNE if given the input of ABNE (i.e, the adjacency matrix), we can efficiently implement sparse access to a matrix such that an estimate of its low-lying spectral density allows us to recover an estimate of the low-lying spectral density of the combinatorial Laplacian. Interestingly, it turns out that we can do so if the input graph is clique-dense (i.e., in precisely the regime that the quantum algorithm for Betti number estimation can efficiently solve). Namely, we can efficiently implement sparse access to the $\binom{n}{k+1} \times \binom{n}{k+1}$-sized matrix Γ_k^G whose columns and rows are indexed by $(k+1)$-subsets of vertices, and whose entries are given by

$$\left(\Gamma_k^G\right)_{i,j} = \begin{cases} (\Delta_k^G)_{i,j} & \text{if } i \text{ and } j \text{ are } (k+1)\text{-cliques,} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

In other words, the entries of the columns and rows that correspond to $(k+1)$-cliques are equal to the corresponding entries of the combinatorial Laplacian, and all other entries are equal to zero. After subtracting the extra nullity caused by adding the $\binom{n}{k+1} - \chi_k$ all-zeros columns and rows, and renormalizing the eigenvalue count by a factor $\binom{n}{k+1}/\chi_k$, the low-lying spectral density of this Γ_k^G is equal to the low-lying spectral density of the combinatorial Laplacian. In Eq. form, we have that

$$N_{\Delta_k^G}(0,b) = \frac{\binom{n}{k+1}}{\chi_k} N_{\Gamma_k^G}(0,b) - \frac{\binom{n}{k+1} - \chi_k}{\chi_k}. \tag{9}$$

From Eq. 9, it is clear that an estimate of $N_{\Gamma_k^G}(0,b)$ up to additive inverse polynomial precision allows us to obtain an estimate of $N_{\Delta_k^G}(0,b)$ up to additive inverse polynomial precision, assuming indeed that the graph is clique-dense – i.e., that $\chi_k/\binom{n}{k+1} \in \Omega\left(1/\mathrm{poly}(n)\right)$.

[4] The exact version of this problem is NP-hard [27]

We emphasize that the above reduction works in precisely the regime where the quantum algorithm for Betti number estimation can efficiently solve ABNE. In other words, LLSD can be used to directly solve the problem that the quantum algorithm for Betti number estimation can efficiently solve. In this regard, LLSD is indeed a generalization of the problem that the quantum algorithm for Betti number estimation can efficiently solve.

Finally, let us discuss the reductions between ABNE and BNE. It is clear that BNE is reducible to ABNE if the size of the smallest nonzero eigenvalue of the combinatorial Laplacian is at least inverse polynomial in n. The reverse direction is unclear, as for BNE the threshold on the eigenvalues is fixed to be exactly zero. A possible approach would be to first project the eigenvalues that lie below the given threshold to zero and then count the zero eigenvalues. However, using techniques inspired by ideas from [28, 29], we have only been able to project these eigenvalues close to zero, as opposed to exactly equal to zero, and we are not aware of any way to circumvent this.

B. The one clean qubit model of computation

In the next section we will show that the complexity of the problems defined in the previous section are closely related to one clean qubit model of quantum computation [30]. In this model we are given a quantum register that is initialized in a state consisting of a single 'clean' qubit in the state $|0\rangle$, and $n-1$ qubits in the maximally mixed state. We can then apply any polynomially-sized quantum circuit to this register, and measure only the first qubit in the computational basis. Following [30], we will refer to the complexity class of problems that can be solved in polynomial time using this model of computation as $\mathsf{DQC1}$ – "deterministic quantum computation with a single clean qubit".

We will refer to a problem as $\mathsf{DQC1}$-hard if any problem in $\mathsf{DQC1}$ can be reduced to it under polynomial time truth-table reductions. That is, a problem L is $\mathsf{DQC1}$-hard if we can solve any problem in $\mathsf{DQC1}$ using polynomially many nonadaptive queries to an oracle for L, together with polynomial time preprocessing of the inputs and postprocessing of the outcomes. Technically, instead of containing the problem of estimating a given quantity up to additive inverse polynomial precision, $\mathsf{DQC1}$ contains the decision problem of deciding whether this quantity is greater than $1/2+\sigma$ or less than $1/2-\sigma$, where σ is some inverse polynomial gap. However, as the estimation versions of these problems are straightforwardly reduced to their decision version using binary search, we will bypass this point from now on and only consider the problems of estimating a given quantity up to inverse polynomial precision [31].

It is widely believed that the one clean qubit model of computation is more powerful than classical computation. For instance, estimating quantities that are supposedly hard to estimate classically, such as the normalized trace of a unitary matrix corresponding to a polynomial-depth quantum circuit and the evaluation of a Jones polynomial at a root of unity, turn out to be complete problems for $\mathsf{DQC1}$ [31]. Moreover, it has been shown that classical computers cannot efficiently sample from the output distribution of the one clean qubit model up to constant total variation distance error, provided that some complexity theoretic conjectures hold [32].

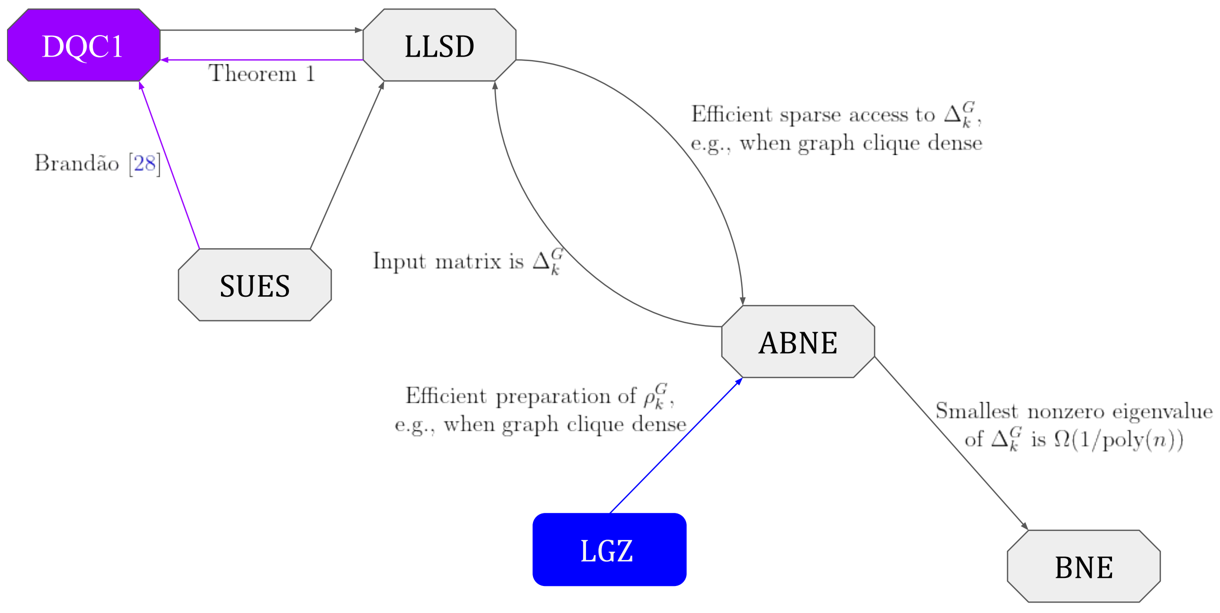
C. Our results

Recall that our goal is to show that the problem that the quantum algorithm for Betti number estimation can efficiently solve is hard for classical computers. In Section III A, we pointed out that the problem that the quantum algorithm for Betti number estimation can efficiently solve is a restriction of ABNE to clique-dense graphs (i.e., graphs which satisfy Eq. 5). Moreover, we showed in Section III A 1 that LLSD is a generalization of this problem. This clearly motivates us to study the classical hardness of LLSD. In this section we present our results, which show that the complexity of LLSD is intimately related to the one clean qubit model.

Our first and main result is that LLSD is hard for the class $\mathsf{DQC1}$, even when the input is restricted to log-local Hamiltonians. As the one clean qubit model of computation is widely believed to be more powerful than classical computation, this shows that LLSD is likely hard for classical computers. We discuss the implications of this result on the classical hardness of the problem that the quantum algorithm for Betti number estimation can efficiently solve in Section III C 1.

Theorem 1. *LLSD is $\mathsf{DQC1}$-hard. Moreover, LLSD with the input restricted to log-local Hamiltonians remains $\mathsf{DQC1}$-hard.*

We will now give a sketch of our proof of the above theorem, the complete proof can be found in the Supplemental Material. The main idea behind the proof is to show that we can use LLSD to estimate a quantity similar to a normalized subtrace, or more precisely, a normalized sum of eigenvalues below a given threshold. As Brandão has shown that estimating this normalized subtrace is $\mathsf{DQC1}$-hard [25], this would prove that LLSD is also $\mathsf{DQC1}$-hard. We estimate this normalized subtrace by constructing a histogram approximation of the low-lying eigenvalues, and afterwards computing the mean of this histogram. To construct this histogram, we use LLSD to estimate the number of eigenvalues that lie in each bin. To avoid double counting of eigenvalues due to imprecisions around the thresholds of the bins, we subtract the output of LLSD with the eigenvalue threshold set to the lower-threshold of the bin from the output of LLSD with the eigenvalue threshold set to the upper-threshold of the bin. By doing so, we obtain an estimate of the number of eigenvalues within the bin, and we misplace eigenvalues



DQC1
LLSD
Theorem 1
Brandão [28]
SUES
Efficient sparse access to Δ_k^G, e.g., when graph clique dense
Input matrix is Δ_k^G
ABNE
Efficient preparation of ρ_k^G, e.g., when graph clique dense
Smallest nonzero eigenvalue of Δ_k^G is $\Omega(1/\text{poly}(n))$
LGZ
BNE

FIG. 3: Overview of the relationships between the problems (in grey), algorithm (in blue) and complexity class (in purple) studied in this paper. The arrows A $\xrightarrow{C}$ B stand for: "A can efficiently solve B if condition C holds".

by at most one bin[1].

Our second result shows that the complexity of LLSD is more closely related to DQC1 than just hardness. Namely, we point out that if the input to LLSD is restricted to log-local Hamiltonians (or more generally, any type of Hamiltonian that allows for efficient Hamiltonian simulation using $\mathcal{O}(\log(n))$ ancilla qubits), then it can be solved using the one-clean qubit model. From this it follows that LLSD is DQC1-complete if the input is restricted to log-local Hamiltonians. The main idea behind why we can solve these instances of LLSD using the one clean qubit model is that the one-clean qubit model can simulate having access to up to $\mathcal{O}(\log(n))$ pure qubits [31]. These pure qubits allow for Hamiltonian simulation techniques based on the Trotter-Suzuki formula [33] and for quantum phase estimation up to the required precision. We summarize this in the following theorem, the proof of which can be found in the Supplemental Material.

Theorem 2. LLSD *with the input restricted to log-local Hamiltonians is* DQC1*-complete.*

As an added result, we find that the complexity of SUES with the input restricted to log-local Hamiltonians is also closely related to DQC1. The complexity of this instance SUES was stated as an open problem by Wocjan and Zhang [24]. Moreover, we believe that it is interesting to study the complexity of SUES, as this problem can potentially find practical applications beyond both LLSD and Betti number estimation. We remark that SUES with the input restricted to log-local Hamiltonians was already shown to be DQC1-hard by Brandão [25]. Here we point out that the complexity of this instance of SUES is more closely related to the one clean qubit model than just hardness, as it can also be solved using $\text{DQC1}_{\log n}$ circuits, that is, DQC1 circuits where we are allowed to measure logarithmically many of the qubits in the computational basis at the end (to read out the encoding of the eigenvalue). The proof of the following proposition can be found in the Supplemental Material.

Proposition 3. SUES *with the input restricted to log-local Hamiltonians can be solved in polynomial time by the one clean qubit model with logarithmically many qubits measured at the end.*

1. *Quantum advantage for topological data analysis*

The results discussed in the previous section are not sufficient to conclude that ABNE or BNE are hard for classical computers. This is in particular true for the problem that the quantum algorithm for Betti number estimation can actually efficiently solve. Nonetheless, because LLSD is a generalization of the problem that the quantum algo-

[1] We believe that our approach could be modified to a Karp reduction by encoding an instance of the normalized subtrace estimation problem into a single instance of LLSD. This would entail manipulating the Kitaev circuit-to-Hamiltonian construction and taking direct sums of matrices. As this reduction is not vital for our claim, we leave this question open for future work.

rithm for Betti number estimation can efficiently solve, our result shows that – aside from the matter regarding the restriction to combinatorial Laplacians – the quantum algorithm for Betti number estimation solves a classically intractable problem which in some cases captures interesting information concerning an underlying graph. Moreover, our result eliminates the possibility of certain routes for dequantization, namely those that are oblivious to the particular structure of the input matrix, which in particular eliminates the approaches of Tang et al. [8].

The remaining open question regarding the classical hardness of ABNE and the problem that the quantum algorithm for Betti number estimation can efficiently solve is whether LLSD remains classically hard when we restrict the input to combinatorial Laplacians of arbitrary and clique-dense graphs, respectively. Even though these restrictions on the input seem quite stringent, note that our result shows that LLSD is already DQC1-hard for the restricted family of log-local Hamiltonians obtained from Kitaev's circuit-to-Hamiltonian construction[1]. We tried to close this gap by investigating whether combinatorial Laplacians already contain such Hamiltonians as submatrices of sufficiently large instances. While indeed various matrices related to quantum gates can be found as submatrices of combinatorial Laplacians, we did not succeed in finding an explicit embedding. In our view, this remains a promising approach to showing that LLSD remains classically hard when restricted to combinatorial Laplacians. Nonetheless, other routes (e.g., by proving classical hardness of LLSD when restricted to other sets of matrices such as 0-1 matrices, or by going through the discrete structures related to Tutte and Jones polynomials [31, 34]) are possible as well.

The remaining open questions regarding the classical hardness of BNE are the same as those regarding the classical hardness of ABNE, except that there is one additional open question. Namely, assuming that ABNE is classically hard, the remaining open question regarding the classical hardness of BNE is whether estimating the number of eigenvalues exactly equal to zero is at least as hard as estimating the number of eigenvalues below a given inverse polynomially small threshold. This question was already discussed in Section III A 1 when we examined the reductions between ABNE and BNE. As discussed there, one approach would be to project the eigenvalues below the given threshold to zero, and afterwards count the zero eigenvalues.

Regardless, even if LLSD does not remain classically hard when restricted to combinatorial Laplacians, we can envision practical generalizations of the quantum-algorithmic methods used by the algorithm for Betti number estimation that go beyond Betti numbers, as we will demonstrate in the next section. Specifically, in the next section we provide efficient quantum algorithms for two concrete examples of such practical generalizations, together with complexity-theoretic evidence of their classical hardness. The first example we discuss is numerical rank estimation, an important problem in machine learning, data analysis and many other applications. The second example is spectral entropy estimation, which can be used as a tool in complex network analysis.

[1] In [25, 35] and our case it is unclear whether this holds for k-local Hamiltonians with constant k, as the standard constructions of these local Hamiltonians involve a clock register that is too large.

IV. QUANTUM SPEEDUPS FOR TOPOLOGICAL DATA ANALYSIS AND BEYOND

In the previous section we provided evidence that the computational problems tackled by the quantum algorithm for Betti number estimation are likely hard for classical computers. However, we have also highlighted how whether the quantum algorithm can efficiently solve the problem related to topological data analysis depends on certain properties of the graph. Here we characterize families of graphs on which the quantum algorithm for Betti number estimation achieves speedups over classical algorithms.

Afterwards, we discuss extensions of the quantum-algorithmic methods behind the algorithm for Betti number estimation that go beyond Betti numbers, to demonstrate that these methods give rise to a potential source of practical quantum advantage based on our hardness result in the previous section. In particular, we provide efficient quantum algorithms for numerical rank estimation (an important problem in machine learning and data analysis) and spectral entropy estimation (which can be used to compare complex networks), together with complexity-theoretic evidence of their classical hardness.

A. Graphs with a quantum advantage

In Section II B we saw that the quantum algorithm for Betti number estimation can efficiently estimate approximate Betti numbers (i.e., efficiently solve the ABNE problem) if the input graph meets certain criteria. Specifically, the graph has to be such that one can either efficiently prepare the maximally mixed state over all its cliques (see Eq. 4), or the zero-padded combinatorial Laplacian (see Eq. 8) has to be sufficiently dense. We have seen that both these criteria are met if the graph is clique-dense, i.e., if it satisfies Eq. 5. In this section, we use the so-called *clique-density theorem* [36] and a result of Moon and Moser [37, 38] to directly characterize families of clique-dense graphs in terms of the required number of edges.

Let us consider clique sizes $k \geq 3$, let $\gamma > \frac{k-2}{2(k-1)}$ be a constant, and let G be any graph on n vertices with at least γn^2 edges. Suppose we want to estimate the k-th approximate Betti number of this graph, where k and the precision parameters are constant. Recall that the quantum algorithm for Betti number estimation solves this in time

$$\widetilde{\mathcal{O}}\left(\sqrt{n^{k+1}/\chi_k}+n^3\right),$$

where χ_k denotes the number of $(k+1)$-cliques in G. The clique density theorem [36] directly guarantees that our graph satisfies

$$\chi_k \in \Omega(n^{k+1}),$$

which is a phenomenon known as "supersaturation". In particular, this implies that our graph is clique-dense and that the quantum algorithm for Betti number estimation can solve this instance of ABNE in time

$$\widetilde{\mathcal{O}}\left(n^3\right).$$

Moreover, as discussed in Section II B 2, the best known classical algorithms require time

$$\mathcal{O}\left(n^{k+1}\right),$$

as the number of nonzero entries of the corresponding combinatorial Laplacian is at least χ_k. We conclude that in these instances the quantum algorithm for Betti number estimation achieves a $(k-2)$-degree polynomial speedup over the best known classical methods, which for large enough k might allow for runtime advantages on prospective fault-tolerant computers, even when all overheads are accounted for as discussed in [11].

We can push the separation between the best known classical algorithm and the quantum algorithm even further. Consider the same setting as above, but with $\gamma = \frac{k-1}{k}$ and we allow k to scale with n. Using a result of Moon and Moser [37–39], we can derive that in this setting the graph satisfies

$$\binom{n}{k+1}/\chi_k \in \mathcal{O}\left(k^k\right).$$

Therefore, the quantum algorithm can estimate the k-th approximate Betti number in time

$$\mathcal{O}\left(k^{2+k/2}+n^3\right).$$

On the other hand, the best known classical algorithm runs in time

$$\mathcal{O}\left(n^{k+1}/k^{2k}\right),$$

as the number of nonzero entries of the corresponding combinatorial Laplacian is at least $\chi_k \geq n^{k+1}/k^{2k}$. Therefore, if k scales with n in an appropriate way, then the quantum algorithm can achieve superpolynomial speedups over the best known classical method. For instance, if $k=\log n$, then the quantum algorithm runs in time $2^{\mathcal{O}(\log n \log\log n)}$, whereas the best known classical algorithm runs in time $2^{\mathcal{O}\left((\log n)^2\right)}$.

The graphs in the above settings are very edge-dense (which occurs in topological data analysis if the grouping-scale ϵ approaches the maximum distance between two datapoints) and it is unknown whether in this regime better classical algorithms are possible. Other settings where superpolynomial speedups are possible would include those where the cliques are not as numerous, but rather there exist efficient ways to enumerate them, or sample from them (e.g., if a large independent analysis was done in a separate pre-processing stage). Another option in this direction is to investigate approximate heuristic clique-sampling methods (as opposed to uniform sampling by rejection sampling) In this case we forego analytic proofs of correctness, though in practice this may be a reasonable option.

B. Numerical rank estimation

In this section we identify a practically important application of the problem of estimating the number of small eigenvalues (which we called LLSD) called *numerical rank estimation*. The numerical rank of a matrix $H \in \mathbb{C}^{2^n\times 2^n}$ is the number of eigenvalues that lie above some given threshold b, i.e., it is defined as

$$r_H(b) = \frac{1}{2^n}\sum_{k\,:\,\lambda_k>b} 1,$$

where $\lambda_1 \leq \cdots \leq \lambda_{2^n-1}$ denote the eigenvalues of H. By the rank-nullity theorem we have that

$$r_H(b) = 1 - N_H(0,b),$$

which shows that we can estimate the numerical rank using low-lying spectral density estimation and that the error scaling is the same.

Many machine learning and data analysis applications deal with high-dimensional matrices whose relevant information lies in a low-dimensional subspace. To be specific, it is a standard assumption that the input matrix is the result of adding small perturbations (e.g., noise in the data) to a low-rank matrix. This turns the input matrix into a high rank matrix, that can be well approximated by a low-rank matrix. Techniques such as principle component analysis [40] and randomized low-rank approximations [41] are able exploit this property of the input matrix. However, these techniques often require as input the dimension of this low-dimensional subspace, which is often unknown. This is where numerical rank estimation comes in, as it can estimate the dimension of the relevant subspace by estimating the number of eigenvalues that lie above the "noise-threshold". In addition, being able to determine whether the numerical rank of a matrix is large or small enables one to assert whether the above low-rank approximation techniques apply at all, or not

From Theorem 1 it directly follows that quantum computers achieve an exponential speedup over classical computers for numerical rank estimation of matrices specified via sparse access (unless the one clean qubit model can be efficiently simulated on a classical computer). Still, it is also interesting to consider settings where the matrix is specified via a different input model. In the remainder of this section we study two examples of different input models. First, motivated by a more practical perspective we consider a seemingly weaker input model that is more closely related to the input models that appear in classical data analysis settings. Second, we consider a likely stronger input model that appears throughout quantum machine learning literature, which is more informative from a complexity-theoretic perspective.

In typical (classical) applications, matrices are generally not specified via sparse access. Here we consider an input model that is more closely related to what is encountered in the classical setting. Specifically, we consider the case where a sparse matrix A of size $2^n \times 2^n$ is specified as a list of triples

$$\left\{(i_k, j_k, A_{i_k,j_k}) \mid A_{i_k,j_k} \neq 0\right\},$$

which is sorted lexicographically by column and then row. Storing matrices in this type of memory structure is very natural when dealing with matrices with a limited number of nonzero entries (which we denote by $\mathsf{N_{nz}}$). Now, for the quantum analogue we consider the same specification but we suppose that it is stored in a QRAM-type memory, only additionally allowing us to query it in superposition as follows:

$$\sum_k \alpha_k \left|k\right\rangle \left|0\right\rangle \mapsto \sum_k \alpha_k \left|k\right\rangle \left|i_k, j_k, A_{(i_k,j_k)}\right\rangle .$$

Since the list is sorted, and since A is sparse, we can still simulate column-wise sparse access in $\mathcal{O}\left(\log \mathsf{N_{nz}}\right)$ queries, essentially by using binary search. Therefore, if A is Hermitian, then the quantum algorithm can estimate its numerical rank in time $\mathcal{O}\left(\mathrm{poly}\left(n, \log \mathsf{N_{nz}}\right)\right)$. On the other hand, the best known classical algorithms run in time $\mathcal{O}\left(\mathsf{N_{nz}}\right)$ [21–23]. Consequently, the quantum algorithm achieves a speedup over the best known classical algorithm if $\mathsf{N_{nz}}$ is at least a high-enough degree polynomial in n (and it achieves an exponential speedup if $\mathsf{N_{nz}}$ is itself exponential). For the case where A is not Hermitian, recall that we also need sparse access to $A^\dagger$. For this issue we found no general method that can do so in time less than $\mathcal{O}\left(\mathsf{N_{nz}}\right)$, without assuming a high sparsity degree. However, this sparsity degree then exactly offsets any potential quantum advantage in the full algorithm complexity.

Next, we consider a likely stronger input model which is widely-studied in the quantum machine learning literature. Specifically, we study the quantum-accessible data structure introduced in [9, 43], which can generate quantum states proportional to the columns of the input matrix, together with a quantum state whose amplitudes are proportional to the 2-norms of the columns. When the input matrix is provided in this quantum-accessible data structure, the quantum-algorithmic methods of [28, 44] can be used to estimate its numerical rank in time $\mathcal{O}\left(\mathrm{poly}(A_{\max}, n)\right)$, where $A_{\max} = \max_{i,j} |A_{ij}|$. The classical analogue of this quantum-accessible data structure is the ℓ^2-sampling access model introduced in [7], which brought forth the “dequantization” methods discussed in [8]. At present it is not clear whether assuming ℓ^2-sampling access allows us to efficiently the estimate numerical rank using dequantization, or other methods. Here both possibilities are interesting. Firstly, if numerical rank estimation remains equally hard with ℓ^2-sampling access, then it shows that quantum algorithms relying on the methods of LGZ have a chance of maintaining their exponential advantage in more general scenarios. Secondly, if an efficient classical algorithm for numerical rank estimation is possible with ℓ^2-sampling access, then this leads to new insights regarding the hardness of the one clean qubit model. Recall that we have shown that estimating the numerical rank of matrices specified via sparse access is $\mathsf{DQC1}$-hard (in the sense that, if a classical algorithm could do so efficiently given analogous access, then it can be used to efficiently solve all problems in $\mathsf{DQC1}$). Now for the sparse matrix case, the only difference between sparse access and ℓ^2-sampling access is that the latter allows you to sample from a distribution whose probabilities are proportional to the 2-norms of the columns. Indeed, the other part (i.e., sampling from distributions whose probabilities are proportional to the squared entries of the columns) is straightforward when the matrix is specified via sparse access. This implies that, if ℓ^2-sampling access allows us to efficiently estimate the numerical rank of sparse matrices, then producing samples according to the 2-norms of the columns of a sparse matrix is $\mathsf{DQC1}$-hard. This also holds for the log-local Hamiltonian setting, so it would also follow that sampling from a distribution proportional to the 2-norms of the columns of log-local Hamiltonians is $\mathsf{DQC1}$-hard.

C. Combinatorial Laplacians beyond Betti numbers

In the previous section we discussed a practical application of the quantum-algorithmic methods behind the algorithm for Betti number estimation by using the same methods, but changing the family of input matrices (i.e., going beyond combinatorial Laplacians). In this section we take a different approach, namely we again consider the combinatorial Laplacians, but investigate applications beyond Betti number estimation relying on different algorithms than the one for low-lying spectral density estimation. Moreover, we will again find regimes where the same type of evidence of classical hardness can be provided, further motivating investigations into quantum algorithms that operate on the combinatorial Laplacians.

The eigenvalues and eigenvectors of the combinatorial Laplacian have many interesting graph-oriented applications beyond the applications in topological data analysis discussed in Section II. The intuition behind this is that the combinatorial Laplacian can be viewed as a generalization of the standard graph Laplacian. For example, there exist generalizations of spectral clustering and label propagation (important techniques in machine learning that are used for dimensionality reduction and classification) which utilize the eigenvalues and eigenvectors of the combinatorial Laplacians [48]. Moreover, the eigenvalues of a normalized version of the combinatorial Laplacian convey information about the existence of circuits of cliques (i.e., ordered lists of adjacent cliques that cover the whole graph) and about the chromatic number [49]. Additionally, Cheeger's inequality – which relates the sparsest cut of a graph to the smallest nonzero eigenvalues of its standard graph Laplacian – turns out to have a generalization that utilizes the combinatorial Laplacian [50]. Lastly, Kirchhoff's matrix tree theorem – which relates the eigenvalues of the standard graph Laplacian to the number of spanning trees – turns out to also have a similar generalization [51].

The specific problem that we study in this section is that of sampling from a distribution over the eigenvalues whose probabilities are proportional to the magnitude of the eigenvalues. In particular, we give a quantum algorithm that efficiently samples from an approximation of these distributions and we show that sampling from these distributions for arbitrary sparse Hermitian matrices is again as hard as simulating the one clean qubit model, showing that it is classically intractable (unless the one clean qubit model can be efficiently simulated on a classical computer). Moreover, we point out how this quantum algorithm can speed up spectral entropy estimation, which when applied to combinatorial Laplacians can be used to compare complex networks.

We define the problem that we study in this section as follows.

Sparse weighted eigenvalue sampling (SWES)
Input:
1) A sparse positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$, with $||H|| \leq 1$ and $\mathrm{Tr}\,(H)\,/2^n \in \mathcal{O}\,(\mathrm{poly}(n))$.
2) An estimation precision $\delta \in \Omega\,(1/\mathrm{poly}(n))$.
3) A sampling error probability $\mu \in \Omega\,(1/\mathrm{poly}(n))$.

Output: A sample drawn from a (δ, μ)-approximation of the distribution $p(\lambda_j) = \lambda_j/\mathrm{Tr}\,(H)$.

Using the subroutines of the quantum algorithm for Betti number estimation (i.e., Hamiltonian simulation and quantum phase estimation), we can efficiently sample from an approximation of the distribution of SWES defined above. In fact, we can efficiently "quantum-sample" from $p(\lambda_j)$, that is, we can implement *purified quantum query-access* [52]. To be precise, we can implement an approximation of the unitary U_H (and its inverse) which acts as

$$U_H \left|0\right\rangle_A \left|0\right\rangle_B = \left|\psi_H\right\rangle = \sum_{j=0}^{2^n-1} \sqrt{p\,(\lambda_j)} \left|\psi_j\right\rangle_A \left|\phi_j\right\rangle_B, \quad (10)$$

such that $\mathrm{Tr}_B\left(\left|\psi_H\right\rangle\left\langle\psi_H\right|\right) = H/\mathrm{Tr}\,(H)$. As we will discuss at the end of this section, purified quantum query-access has been shown to be more powerful than standard classical sampling access, as it can speedup the postprocessing of the samples when trying to find out properties of the underlying distribution [52].

We implement an approximation of the purified quantum-query access defined in Eq. 10 as follows:

1. Prepare the following input state by taking a maximally entangled state (which can always be expressed in the eigenbasis of H in one of its subsystems) and adding two ancillary registers

$$\left|\psi\right\rangle_{in} = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|0^t\right\rangle \otimes \left|0\right\rangle_{flag},$$

where $\{\left|\psi_k\right\rangle\}_{k=0}^{2^n-1}$ are orthonormal eigenvectors of H and $\{\left|\phi_k\right\rangle\}_{k=0}^{2^n-1}$ is an orthonormal basis of $\mathbb{C}^{2^n}$.

2. Use Hamiltonian simulation on H, and apply quantum phase estimation of the realized unitary to the first register to prepare the state

$$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^t} \alpha_{k,j} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_{k,j}}\right\rangle \otimes \left|0\right\rangle_{flag}$$
$$\approx \frac{1}{\sqrt{N}} \sum_{k=0}^{2^n-1} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_k}\right\rangle \otimes \left|0\right\rangle_{flag},$$

where the $\widetilde{\lambda_{k,j}}$ are t-bit strings, $|\alpha_{k,j}|^2$ is close to 1 if and only if $\lambda_k \approx \widetilde{\lambda_{k,j}}$, and $\widetilde{\lambda_k}$ denotes the best t-bit approximation of λ_k.

3. Use controlled rotations to "imprint" the t-bit approximations of the eigenvalues into the amplitudes of the flag-register to prepare the state

$$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^t} \alpha_{k,j} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_{k,j}}\right\rangle \otimes \left(\sqrt{\widetilde{\lambda_{k,j}}} \left|0\right\rangle_{flag} + \sqrt{1-\widetilde{\lambda_{k,j}}} \left|1\right\rangle_{flag}\right)$$
$$\approx \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_k}\right\rangle \otimes \left(\sqrt{\widetilde{\lambda_k}} \left|0\right\rangle_{flag} + \sqrt{1-\widetilde{\lambda_k}} \left|1\right\rangle_{flag}\right).$$

4. Use fixed point amplitude amplification to amplify states whose flag-register is in the state $\left|0\right\rangle$ to prepare an approximation of the state

$$\frac{1}{\sqrt{\mathrm{Tr}(H)}} \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^t} \alpha_{k,j} \sqrt{\widetilde{\lambda_{k,j}}} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_{k,j}}\right\rangle \otimes \left|0\right\rangle_{flag}$$
$$\approx \frac{1}{\sqrt{\mathrm{Tr}(H)}} \sum_{k=0}^{2^n-1} \sqrt{\widetilde{\lambda_k}} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_k}\right\rangle \otimes \left|0\right\rangle_{flag}.$$

5. Finally, uncompute and discard the eigenvalue- and flag-register to prepare the state

$$\begin{aligned}|\psi_H\rangle &= \frac{1}{\sqrt{\mathrm{Tr}(H)}}\sum_{k=0}^{2^n-1}\sum_{j=0}^{2^t}\alpha_{k,j}\sqrt{\widetilde{\lambda_{k,j}}}\,|\psi_k\rangle\,|\phi_k\rangle \\ &\approx \frac{1}{\sqrt{\mathrm{Tr}(H)}}\sum_{k=0}^{2^n-1}\sqrt{\widetilde{\lambda_k}}\,|\psi_k\rangle\,|\phi_k\rangle\,.\end{aligned}$$

Looking at the cost of the above algorithm, we note that Steps 2 and 3 can be implemented up to polynomial precision in time $\mathcal{O}\left(\mathrm{poly}(n)\right)$. Also, note that Step 4 can be implemented up to polynomial precision in time $\mathcal{O}\left(\sqrt{2^n/\mathrm{Tr}\left(H\right)}\right)$, which brings the total runtime to

$$\mathcal{O}\left(\mathrm{poly}(n)+\sqrt{2^n/\mathrm{Tr}\left(H\right)}\right).$$

Besides being able to efficiently sample from an approximation of SWES on a quantum computer, we show that SWES requires exponential time on a classical computer (unless the one clean qubit model can be efficiently simulated on a classical computer). To be precise, we show that sampling from SWES allows us to efficiently estimate the normalized subtrace discussed in Section III C, which is known to be DQC1-hard [25]. We gather this in the following theorem, the proof of which can be found in the Supplementary Material.

Theorem 4. *SWES is DQC1-hard. Moreover, SWES with the input restricted to log-local Hamiltonians remains DQC1-hard.*

The above theorem motivates us to look for practical applications of SWES, or more specifically, of the purified quantum query-access described in Eq. 10. We end this section by discussing such an application called spectral entropy estimation, which when applied to combinatorial Laplacians can be used to compare complex networks. We note that the classical hardness of SWES opens up another road towards practical quantum advantage, as it could be that combinatorial Laplacians arising in complex network analysis form a rich enough family for which SWES remains classically hard when restricted to them.

1. Spectral entropy estimation of the combinatorial Laplacian

Recently, several quantum information-inspired entropic measures for complex network analysis have been proposed [53, 54]. One example of these are spectral entropies of the combinatorial Laplacian, which measure the degree of overlapping of cliques within the given complex network [55–57].

If $\lambda_0,\ldots,\lambda_{d_k^G-1}$ denote the eigenvalues of a combinatorial Laplacian Δ_k^G (i.e., $d_k^G=\dim\mathcal{H}_k^G$), then its *spectral entropy* is defined by

$$S(\Delta_k^G)=-\sum_{j=0}^{d_k^G-1}p(\lambda_j)\log(p(\lambda_j)),\tag{11}$$

where we define $p(\lambda_j)=\lambda_j/\left(\sum_k\lambda_k\right)$. This spectral entropy coincides with the von Neumann entropy of $\Delta_k^G/\mathrm{Tr}\left(\Delta_k^G\right)$. Equivalently, it coincides with the Shannon entropy of the distribution $p(\lambda_j)$. Another entropy that is used in complex network analysis is the *α-Renyi spectral entropy*, which is given by

$$S_\alpha(\Delta_k^G)=\frac{1}{1-\alpha}\log\left(\sum_{j=0}^{d_k-1}p(\lambda_j)^\alpha\right),\tag{12}$$

where $\alpha\geq 0$ and $\alpha\neq 1$. The limit for $\alpha\to 1$ is the spectral entropy as defined in Eq. 11.

To estimate the spectral entropy defined in Eq. 11, one can use techniques from [58, 59] to classically postprocess samples from $p(\lambda_j)$ that one obtains from the quantum algorithm for SWES described in the previous section. However, as we can in fact implement purified quantum query-access using the algorithm described in the previous section, the postprocessing can be quadratically sped using quantum methods [52]. The same idea of speeding up the postprocessing of samples by using quantum methods also holds for the α-Renyi entropy defined in Eq. 12, where one can either classically postprocess the samples [60], or use quantum algorithms to do so faster [61].

Because we have shown that sampling from SWES is DQC1-hard, the above approach to spectral entropy estimation can not be done efficiently on a classical computer – i.e., it cannot be dequantized – when generalized to arbitrary sparse matrices (unless the one clean qubit model can be efficiently simulated on a classical computer). Moreover, as the α-Renyi entropy is the logarithm of the Schatten p-norm, and it is known that estimating Schatten p-norms is DQC1-hard [35], we find that computing the α-Renyi entropies is classically hard in general (again, unless the one clean qubit model can be efficiently simulated on a classical computer).

V. POSSIBILITIES AND CHALLENGES FOR IMPLEMENTATIONS

As near-term quantum devices are still limited, it is crucial to make sure to use them to their fullest potential when implementing a quantum algorithm. Near-term devices are limited in size, gates are error prone, qubits decohere, and their architectures are limited [13]. We are therefore interested in algorithms that are shallow and require few gates (to minimize the effect of decoherence and overall gate errors), which are not too demanding regarding connectivity, while achieving advantages with few qubits and being tolerant to noise (which will inevitably be present in the system regardless of the depth and gate count). The quantum algorithms we consider use Hamiltonian simulation and quantum phase estimation. Fortunately, both resource optimization [62] and error-mitigation [63–67] for these routines are important topics for the broadly investigated field of quantum algorithms for quantum chemistry and many-body physics, and any progress achieved for those purposes can be readily applied. In this section we will focus on the issues of size and noise. First, we investigate the required number of qubits and we propose methods on how to reduce this. Based on these methods, we provide a brief estimate of the number of qubits required to challenge classical methods. Finally, we discuss the robustness of the algorithm to noise in the hardware.

To analyze the number of qubits required to implement Hamiltonian simulation of a $2^n \times 2^n$-sized input matrix, we consider two possible scenarios: the input matrix is either given to us as local terms, or it is specified via sparse access. If the input matrix is given to us as local terms, then we can implement Hamiltonian simulation based on the Trotter-Suzuki formula [33]. As this Hamiltonian simulation technique does not require ancillary qubits (assuming the available gate set can implement each of the Trotter steps without ancillary qubits) [35], we can implement it using n qubits. On the other hand, if the input matrix specified via sparse access, then we have to use more complex Hamiltonian simulation techniques (e.g., based on quantum signal processing [19]). The downside of these methods is that they require an ancillary register to 'load' the queries to the sparse-access oracles onto. By having to add this ancillary register, the total number of qubits required to implement these Hamiltonian simulation techniques becomes $2n + r + 1$, where r is the number of bits used to specify the entries of the input matrix. So, sparse-access oracles more than double the required number of qubits.

Thus, when possible it is advisable to avoid using sparse access when having first proof-of-principle demonstrations of quantum advantage in mind, which we propose can be done as follows. Specifically, it is possible to trade-off the required number of ancilla qubits for some amount of precompilation and some extra depth of the precompiled circuit, in the following two ways. First, one could decompose the input matrix in terms of a linear combination of unitaries, and use related techniques for Hamiltonian simulation of such input matrices [68]. This brings the required number of qubits down from $2n + r + 1$ to $n + \log(m)$, where m is the number of terms in the linear combination of unitaries. Secondly, one could decompose the input matrix in terms of a sum of local Hamiltonians and use Hamiltonian simulation based on the Trotter-Suzuki formula. This brings the required number of qubits down from $2n + r + 1$ to n. Thus, both approaches can halve the number of required qubits, however, one has to be careful as finding such decompositions may constitute a dominating overhead.

In case of Betti number estimation, we note that both of the above approaches to precompilation are in fact feasible and meaningful. Namely, in this case the input matrix (i.e., the Dirac operator) is fixed for a given graph size. This implies that we can estimate the Betti numbers of exponentially many graphs (i.e., all graphs of that given size) using a single circuit that implements Hamiltonian simulation of the corresponding Dirac operator. In other words, this precompilation needs to be done once for each data size, but not for each specific dataset. Besides the ability to bring down the required number of qubits, these approaches are also interesting from a different perspective. Namely, they may shed light on what unitaries occur as submatrices of the Dirac operator and combinatorial Laplacians. This could lead to new avenues to explore regarding the hardness of Betti number estimation, as it could offer the possibility of identifying an explicit way of encoding matrices that arise from Kitaev's circuit-to-Hamiltonian construction into the structure of the Dirac operator and combinatorial Laplacians.

Next, we focus on the number of qubits required for the quantum phase estimation. Standard quantum phase estimation requires an eigenvalue register of t qubits, where t denotes the desired bits of precision for the eigenvalue estimates (which consequently determines the threshold in low-lying spectral density estimation). Fortunately, much improvement is possible in terms of the size of this eigenvalue register. First, as low-lying spectral density is only concerned with whether the t-bit approximation of an eigenvalue is zero or not, we can bring the size the of eigenvalue register down to $\log(t)$ by using a counter [69]. Moreover, we can bring the size of this eigenvalue register down to a single qubit at the expense of classical post-processing and qubit reinitialization methods [70, 71].

We can now give the brief estimate of the number of qubits needed for demonstrations of quantum advantage (i.e., sizes needed to go beyond the best known classical methods). The best known classical methods for low-lying spectral density estimation, to our knowledge, are able to estimate the rank of a matrix in time linear in the number of nonzero entries [21–23]. These methods are at most quadratically faster than exact diagonalization, which tends to hit a practical wall around matrices of size 2^{40}. We therefore look at how many qubits are required to estimate the low-lying spectral density below a threshold of about 10^{-9} (i.e., $t \approx \log(10^9) < 30$) of matrices of size around 2^{80} (i.e., $n \approx 80$). In this case, the required number of qubits for standard implementations is approximately

$$2n + r + 1 + t \approx 200,$$

when we are for instance interested in matrices such as the combinatorial Laplacians. If we precompile the input matrix through finding a decomposition in terms of local Hamiltonians, this can be reduced to

$$n + t \approx 110.$$

This can be further reduced to $n + \log(t)$ by using a counter in the eigenvalue register. Lastly, by using a single-qubit eigenvalue register (at the cost of classical postprocessing and qubit reinitialization) we bring the number of required qubits in the optimal case down to

$$n + 1 \approx 80,$$

which is tantalizingly close to what leading teams are expected to achieve in the immediate future in terms of qubit numbers alone.

When it comes to the robustness to noise in the hardware, we need to consider the type of algorithm that is being applied (i.e., how noise affects this algorithm in general) together with the specifics of the application. The algorithm we consider involves many iterations of Hamiltonian simulation and quantum phase estimation, where we are interested in the expected value of a two outcome measurement (designating the zero eigenvalues). As noted earlier, these routines are also crucial for quantum algorithms for quantum chemistry and many-body physics, and consequently, all error-mitigation methods developed for these purposes can be readily applied [63–67]. However, as in quantum chemistry and many-body physics one extracts the entire eigenvalues, as opposed to just the frequency of the zero eigenvalue, the application we consider is even less demanding. Additional robustness properties van be inferred from the nature of the particular problem solved. For instance, in machine learning and data analysis applications, the fact that the algorithm serves the purpose of dealing with noise in the data might make noise in the hardware less detrimental compared to when solving more exact problems [1].

Unfortunately, this argument cannot be as readily applied to Betti number estimation, as noise in the data does not corresponds to small perturbations of the simulated matrix (i.e., the combinatorial Laplacian), but rather to a completely different matrix altogether. In turn, small perturbations of the simulated matrix do not corresponds to any meaningful perturbation of the input data. However, we can still identify certain robust features by considering what perturbations of the combinatorial Laplacian entail for the final output, i.e., the low-lying spectral density. Specifically, if the combinatorial Laplacian is perturbed by a small enough matrix (e.g., in terms of operator norm or rank), then the low-lying spectral density remains largely unchanged as such perturbations will not push the low-lying eigenvalues above the threshold. These settings are often studied in the field of perturbation theory [73, 74], which allows us to make these arguments completely formal. In addition, as a random matrix is likely of full rank [75], the perturbed combinatorial Laplacian is also likely of full rank, showing that in the noisy setting we should focus on approximate Betti number estimation methods, as opposed to exact ones. Furthermore, there has already been some work done verifying the robustness of the quantum algorithm for Betti number estimation experimentally [76].

VI. SUMMARY

In this paper we investigated the potential of a class of problems arising from the quantum algorithm for topological data analysis of LGZ [12] to become genuinely useful applications of unrestricted, or even near-term, quantum computers with a superpolynomial speedup over classical computers. We showed that this algorithm along with a number of new algorithms provided by us (with applications in numerical linear algebra, machine learning and complex network analysis) solve problems that are classically intractable under widely-believed complexity-theoretic assumptions by showing that they are as hard as simulating the one clean qubit model. While the complete resolution of the hardness of the topological data analysis problem will require future research into the properties of the combinatorial Laplacians (which as we showed is also interesting for other applications such as complex network analysis), our results eliminate the possibility of generic dequantization methods that are oblivious to the structure of the combinatorial Laplacian. Specifically, our results showed that the methods of the algorithm of LGZ withstand the sweeping dequantization results of Tang et al. [7, 8]. Regarding near-term implementations, we investigated the required resources to challenge the best known classical methods, we identified that implementing sparse access to the input matrix is a major bottleneck in terms of the required number of qubits, and we proposed multiple methods to circumvent this bottleneck via classical precompilation strategies.

In summary, our results show that the quantum-algorithmic methods behind the algorithm of LGZ give rise to a source of both useful and guaranteed exponential quantum speedups (that are amenable to near-term restricted quantum computers), recovering some of the potential for linear-algebraic quantum machine learning to become a killer application of quantum computers.

ACKNOWLEDGMENTS

The authors thank Dorit Aharonov, Tomoyuki Morimae and Ronald de Wolf for early discussions on the topic. The authors are grateful to Ronald de Wolf for careful reading of the manuscript and for giving valuable comments.

This work was supported by the Dutch Research Council (NWO/OCW), as part of the Quantum Software Consortium programme (project number 024.003.037), and through QuantERA project QuantAlgo 680-91-034.

[1] V. Dunjko and P. Wittek, Quantum Views **4**, 32 (2020).
[2] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature **549**, 195 (2017), `arXiv:1611.09347`.
[3] A. W. Harrow, A. Hassidim, and S. Lloyd, Physical review letters **103**, 150502 (2009), `arXiv:0811.3171`.
[4] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Nature **567**, 209 (2019), `arXiv:1804.11326`.
[5] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, Physical Review A **101**, 032308 (2020), `arXiv:1804.00633`.
[6] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Quantum Science and Technology **4**, 043001 (2019), `arXiv:1906.07682`.
[7] E. Tang, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (2019) pp. 217–228, `arXiv:1807.04271`.
[8] N.-H. Chia, A. Gilyén, T. Li, H.-H. Lin, E. Tang, and C. Wang, in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (to appear 2020) `arXiv:1910.06151`.
[9] I. Kerenidis and A. Prakash, in *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)* (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017) `arXiv:1603.08675`.
[10] S. Lloyd, M. Mohseni, and P. Rebentrost, Nature Physics **10**, 631 (2014), `arXiv:1307.0401`.
[11] R. Babbush, J. McClean, C. Gidney, S. Boixo, and H. Neven, arXiv preprint arXiv:2011.04149 (2020).
[12] S. Lloyd, S. Garnerone, and P. Zanardi, Nature communications **7**, 1 (2016), `arXiv:1408.3106`.
[13] J. Preskill, Quantum **2**, 79 (2018), `arXiv:1801.00862`.
[14] R. Ghrist, Bulletin of the American Mathematical Society **45**, 61 (2008).
[15] B. Eckmann, Commentarii Mathematici Helvetici **17**, 240 (1944).
[16] J. Friedman, Algorithmica **21**, 331 (1998).
[17] K. W. Govek, V. S. Yamajala, and P. G. Camara, PLoS computational biology **15**, e1007509 (2019).
[18] S. Gunn and N. Kornerup, arXiv preprint arXiv:1906.07673 (2019).
[19] G. H. Low and I. L. Chuang, Physical review letters **118**, 010501 (2017), `arXiv:1606.02685`.
[20] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. (Cambridge University Press, 2011).
[21] H. Y. Cheung, T. C. Kwok, and L. C. Lau, Journal of the ACM (JACM) **60**, 1 (2013), `arXiv:1203.6705`.
[22] E. Di Napoli, E. Polizzi, and Y. Saad, Numerical Linear Algebra with Applications **23**, 674 (2016), `arXiv:1308.4275`.
[23] L. Lin, Y. Saad, and C. Yang, SIAM review **58**, 34 (2016), `arXiv:1308.5467`.
[24] P. Wocjan and S. Zhang, arXiv preprint quant-ph/0606179 (2006).
[25] F. G. Brandão, *Entanglement theory and the quantum simulation of many-body physics*, Ph.D. thesis, University of London (2008), `arXiv:0810.0026`.
[26] B. Brown, S. T. Flammia, and N. Schuch, Physical review letters **107**, 040501 (2011), `arXiv:1010.3060`.
[27] M. Adamaszek and J. Stacho, Computational Geometry **57**, 8 (2016).
[28] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (2019) pp. 193–204, `arXiv:1806.01838`.
[29] A. Y. Kitaev, A. Shen, M. N. Vyalyi, and M. N. Vyalyi, *Classical and quantum computation*, 47 (American Mathematical Soc., 2002).
[30] E. Knill and R. Laflamme, Physical Review Letters **81**, 5672 (1998), `arXiv:quant-ph/9802037`.
[31] P. W. Shor and S. P. Jordan, Quantum Information & Computation **8**, 681 (2008), `arXiv:0707.2831`.
[32] T. Morimae, Physical Review A **96**, 040302 (2017), `arXiv:1704.03640`.
[33] S. Lloyd, Science , 1073 (1996).
[34] H. Ahmadi and P. Wocjan, Journal of Knot Theory and its Ramifications **19**, 727 (2010).
[35] C. Cade and A. Montanaro, in *13th Conference on the Theory of Quantum Computation, Communication and Cryptography* (2018) `arXiv:1706.09279`.
[36] C. Reiher, Annals of Mathematics , 683 (2016), `arXiv:1212.2454`.
[37] J. Moon and M. L., Publ. Math. Inst. Hung. Acad. Sci. , 283 (1962).
[38] L. Lovász *et al.*, Current Developments in Mathematics **2008**, 67 (2009).
[39] J. Ugander, L. Backstrom, and J. Kleinberg, in *Proceedings of the 22nd international conference on World Wide Web* (2013) pp. 1307–1318, `arXiv:1304.1548`.
[40] I. T. Jolliffe, in *Principal component analysis* (Springer, 1986) pp. 129–155.
[41] N. Halko, P.-G. Martinsson, and J. A. Tropp, SIAM review **53**, 217 (2011), `arXiv:0909.4061`.
[42] S. Aaronson, Nature Physics **11**, 291 (2015).
[43] I. Kerenidis and A. Prakash, Physical Review A **101**, 022316 (2020), `arXiv:1704.04992`.
[44] S. Chakraborty, A. Gilyén, and S. Jeffery, in *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)* (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019) `arXiv:1804.01973`.
[45] A. Gilyén, S. Lloyd, and E. Tang, arXiv preprint arXiv:1811.04909 (2018).
[46] N.-H. Chia, H.-H. Lin, and C. Wang, arXiv preprint arXiv:1811.04852 (2018).
[47] N.-H. Chia, T. Li, H.-H. Lin, and C. Wang, arXiv preprint arXiv:1901.03254 (2019).
[48] B. Osting, S. Palande, and B. Wang, arXiv preprint arXiv:1708.08436 (2017).
[49] D. Horak and J. Jost, Advances in Mathematics **244**, 303 (2013), `arXiv:1105.2712`.
[50] A. Gundert and M. Szedláky, Journal of Computational Geometry **6** (2015), `arXiv:1401.2290`.
[51] A. Duval, C. Klivans, and J. Martin, Transactions of the American Mathematical Society **361**, 6073 (2009), `arXiv:0802.2576`.
[52] A. Gilyén and T. Li, in *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)* (Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020) `arXiv:1902.00814`.
[53] J. Biamonte, M. Faccin, and M. De Domenico, Commu-

nications Physics **2**, 1 (2019), `arXiv:1702.08459`.

[54] M. De Domenico and J. Biamonte, Physical Review X **6**, 041062 (2016), `arXiv:1609.01214`.

[55] F. Passerini and S. Severini, International Journal of Agent Technologies and Systems (IJATS) **1**, 58 (2009), `arXiv:0812.2597`.

[56] D. Simmons, J. Coon, and A. Datta, Journal of Complex Networks **6**, 859 (2018), `arXiv:1707.07906`.

[57] S. Maletić and M. Rajković, The European Physical Journal Special Topics **212**, 77 (2012).

[58] J. Acharya, I. Issa, N. V. Shende, and A. B. Wagner, in *2019 IEEE International Symposium on Information Theory (ISIT)* (IEEE, 2019) pp. 3012–3016, `arXiv:1711.00814`.

[59] G. Valiant and P. Valiant, in *Proceedings of the forty-third annual ACM symposium on Theory of computing* (2011) pp. 685–694.

[60] J. Acharya, A. Orlitsky, A. T. Suresh, and H. Tyagi, IEEE Transactions on Information Theory **63**, 38 (2016), `arXiv:1408.1000`.

[61] S. Subramanian and M.-H. Hsieh, arXiv preprint arXiv:1908.05251 (2019).

[62] B. Bauer, S. Bravyi, M. Motta, and G. K. Chan, arXiv preprint arXiv:2001.03685 (2020).

[63] K. Temme, S. Bravyi, and J. M. Gambetta, Physical review letters **119**, 180509 (2017), `arXiv:1612.02058`.

[64] X. Bonet-Monroig, R. Sagastizabal, M. Singh, and T. O'Brien, Physical Review A **98**, 062339 (2018), `arXiv:1807.10050`.

[65] S. Endo, S. C. Benjamin, and Y. Li, Physical Review X **8**, 031027 (2018), `arXiv:1712.09271`.

[66] S. McArdle, X. Yuan, and S. Benjamin, Physical review letters **122**, 180501 (2019), `arXiv:1807.02467`.

[67] T. E. O'Brien, S. Polla, N. C. Rubin, W. J. Huggins, S. McArdle, S. Boixo, J. R. McClean, and R. Babbush, arXiv preprint arXiv:2010.02538 (2020).

[68] D. W. Berry, A. M. Childs, and R. Kothari, in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science* (IEEE, 2015) pp. 792–809, `arXiv:1501.01715`.

[69] M. Rennela, A. Laarman, and V. Dunjko, arXiv preprint arXiv:2007.07040 (2020).

[70] T. E. O'Brien, B. Tarasinski, and B. Terhal, New Journal of Physics (2019), `arXiv:1809.09697`.

[71] R. D. Somma, New Journal of Physics **21**, 123025 (2019), `arXiv:1907.11748`.

[72] S. Ubaru, Y. Saad, and A.-K. Seghouane, Neural computation **29**, 1317 (2017), `arXiv:1608.05754`.

[73] T. Kato, *Perturbation theory for linear operators*, Vol. 132 (Springer Science & Business Media, 2013).

[74] R. Bhatia, *Matrix analysis*, Vol. 169 (Springer Science & Business Media, 2013).

[75] X. Feng and Z. Zhang, Applied mathematics and computation **185**, 689 (2007).

[76] H.-L. Huang, X.-L. Wang, P. P. Rohde, Y.-H. Luo, Y.-W. Zhao, C. Liu, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan, Optica **5**, 193 (2018), `arXiv:1801.06316`.

Supplemental Material – Towards Quantum Advantage via Topological Data Analysis

I. LLSD IS DQC1-HARD

Following the definition of [1], for any problem $L \in \mathsf{DQC1}$ and every $x \in L$, there exists a quantum circuit U of depth $T \in \mathcal{O}\left(\mathrm{poly}(|x|)\right)$ that operates on $n \in \mathcal{O}\left(\mathrm{poly}(|x|)\right)$ qubits such that

- $x \in L_{yes} \implies p_0 \geq \frac{1}{2} + \frac{1}{\mathrm{poly}(|x|)}$,
- $x \in L_{no} \implies p_0 \leq \frac{1}{2} - \frac{1}{\mathrm{poly}(|x|)}$,

where $p_0 = \mathrm{Tr}\left[(\left|0\right\rangle\left\langle 0\right| \otimes I)U\rho U^\dagger\right]$ and $\rho = \left|0\right\rangle\left\langle 0\right| \otimes I/2^{n-1}$. From this it can be gathered that if we can estimate p_0 to within $1/\mathrm{poly}(|x|)$ additive precision, then we can solve L.

For a positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$ and a threshold $b \in \mathbb{R}_{\geq 0}$, we define the *normalized subtrace* of H up to b as

$$\overline{\mathrm{Tr}_b}(H) = \frac{1}{2^n} \sum_{0 \leq \lambda_k \leq b} \lambda_k,$$

where $\lambda_0 \leq \cdots \leq \lambda_{2^n-1}$ denote the eigenvalues of H. The following result by Brandão shows that if we can estimate the normalized subtrace $\overline{\mathrm{Tr}_b}$ of log-local Hamiltonians up to additive inverse polynomial precision, then we can solve any problem in $\mathsf{DQC1}$. In other words, estimating $\overline{\mathrm{Tr}_b}$ of log-local Hamiltonians up to additive inverse polynomial precision is $\mathsf{DQC1}$-hard.

Proposition 1 (Brandão [2]). *Given as input a description of an n-qubit quantum circuit U of depth $T \in \mathcal{O}\left(poly(n)\right)$ together with a polynomial $r(n)$, one can efficiently construct a log-local Hamiltonian $H \in \mathbb{C}^{T2^n \times T2^n}$ and a threshold $b \in \mathcal{O}\left(poly(n)\right)$ such that*

$$\left|\overline{\mathrm{Tr}_b}(H) - p_0\right| \leq \frac{1}{r(n)}, \tag{1}$$

where $p_0 = \mathrm{Tr}\left[(\left|0\right\rangle\left\langle 0\right| \otimes I)U\rho U^\dagger\right]$ and $\rho = \left|0\right\rangle\left\langle 0\right| \otimes I/2^{n-1}$. Moreover, H also satisfies:

(i) H is positive semidefinite.

(ii) There exists a $\delta \in \Omega\left(1/poly(n)\right)$ such that H has no eigenvalues in the interval $[b, b+\delta]$.

Remark. *The Hamiltonian in the above proposition is obtained by applying Kitaev's circuit-to-Hamiltonian construction directly to the circuit U, but only constraining the input and output of the clean qubit while leaving the other qubits unconstrained (emulating the maximally mixed state).*

We will show that we can efficiently estimate the normalized subtrace $\overline{\mathrm{Tr}_b}$ in Equation 1 to within additive inverse polynomial precision using an oracle for LLSD. To be precise, we show that we can estimate this normalized subtrace to within additive inverse polynomial precision using a polynomial amount of nonadaptive queries to an oracle for LLSD (whose input is restricted to log-local Hamiltonians), together with polynomial-time classical preprocessing of the inputs and postprocessing of the outputs. In other words, we provide a polynomial-time truth-table reduction from the problem of estimating $\overline{\mathrm{Tr}_b}$ to LLSD. We gather this in Lemma 2, which together with Proposition 1 shows that LLSD with the input restricted to log-local Hamiltonians is $\mathsf{DQC1}$-hard under polynomial-time truth-table reductions.

Lemma 2. *Given as input $H \in \mathbb{C}^{T2^n \times T2^n}$ and $b \in \mathcal{O}\left(poly(n)\right)$ as described in Proposition 1, together with a polynomial $q(n)$, one can compute a quantity Λ that satisfies*

$$\left|\Lambda - \overline{\mathrm{Tr}_b}(H)\right| \leq \frac{1}{q(n)},$$

using a polynomial number of queries to an oracle for LLSD, together with polynomial-time classical preprocessing of the inputs and postprocessing of the outputs.

Proof. Define $\Delta = (3q(n))^{-1}$, $M = b/\Delta$, $\epsilon = (6Mbq(n))^{-1}$ and let $\delta < \Delta/3$ be such that H has no eigenvalues in the interval $[b, b+\delta]$. Also, define the thresholds $x_j = (j+1)\Delta$, for $j = 0, \ldots, M-1$. Next, denote by $\hat{\chi}_j$ the outcome of LLSD with threshold $b = x_j$ and precision parameters δ, ϵ as defined above. That is, $\hat{\chi}_j$ is an estimate of $\hat{y}_j$ to within additive accuracy ϵ, where

$$\hat{y}_j = N_H(0, x_j) + \hat{\gamma}_j, \text{ with } 0 \leq \hat{\gamma}_j \leq N_H(x_j, x_j + \delta).$$

Subsequently, define $\chi_0 = \hat{\chi}_0$, $y_0 = \hat{y}_0$ and

$$y_j = \hat{y}_j - \hat{y}_{j-1}, \tag{2}$$

$$\chi_j = \hat{\chi}_j - \hat{\chi}_{j-1}, \tag{3}$$

for $1 \leq j \leq M-1$. Finally, define the estimate

$$\Lambda = \sum_{j=0}^{M-1} \chi_j x_j. \tag{4}$$

We will show that Λ is indeed an estimate of $\overline{\mathrm{Tr}_b}(H)$ to within additive precision $\pm 1/q(n)$. To do so, we define $\gamma_0 = \hat{\gamma}_0$ and $\gamma_j = \hat{\gamma}_j - \hat{\gamma}_{j-1}$ for $1 \leq j \leq M-1$, and we define and expand

$$\Gamma = \sum_{j=0}^{M-1} y_j x_j = \sum_{j=0}^{M-1} \left(N_H(x_{j-1}, x_j) + \gamma_j\right) x_j = \underbrace{\sum_{j=0}^{M-1} N_H(x_{j-1}, x_j) x_j}_{\mathcal{B}:=} + \underbrace{\sum_{j=0}^{M-1} \gamma_j x_j}_{\mathcal{E}_{bin}:=}.$$

We start by upper-bounding the magnitude of the $\mathcal{E}_{bin}$ term. To do so, we rewrite

$$\begin{aligned}
\mathcal{E}_{bin} &= \sum_{j=0}^{M-1} \gamma_j x_j = \hat{\gamma}_0 x_0 + \sum_{j=1}^{M-1} (\hat{\gamma}_j - \hat{\gamma}_{j-1}) x_j \\
&= \sum_{j=0}^{M-1} \hat{\gamma}_j x_j - \sum_{j=1}^{M-1} \hat{\gamma}_{j-1} x_j \\
&= \sum_{j=0}^{M-1} \hat{\gamma}_j x_j - \sum_{j=1}^{M-1} \hat{\gamma}_{j-1} (x_{j-1} + \Delta) \\
&= \sum_{j=0}^{M-1} \hat{\gamma}_j x_j - \sum_{j=1}^{M-1} \hat{\gamma}_{j-1} x_{j-1} - \Delta \sum_{j=1}^{M-1} x_{j-1} \\
&= \underbrace{\hat{\gamma}_{M-1}}_{=0} x_{M-1} - \Delta \underbrace{\sum_{j=1}^{M-1} \hat{\gamma}_{j-1}}_{\leq 1},
\end{aligned}$$

and we conclude that $|\mathcal{E}_{bin}| \leq \Delta$. Next, we upper-bound the absolute difference of $\mathcal{B}$ and $\overline{\mathrm{Tr}_b}(H)$.

$$\left|\mathcal{B} - \overline{\mathrm{Tr}_b}(H)\right| = \left|\sum_{j=0}^{M-1} N_H(x_{j-1}, x_j) x_j - \overline{\mathrm{Tr}_b}(H)\right| \leq \sum_{j=0}^{M-1} \Delta \cdot N_H(x_{j-1}, x_j) \leq \Delta.$$

Finally, we upper-bound the absolute difference between Λ and Γ.

$$|\Lambda - \Gamma| = \left|\sum_{j=0}^{M-1} (\chi_j - y_j) x_j\right| \leq \left|\sum_{j=0}^{M-1} 2\epsilon x_j\right| \leq M \cdot 2\epsilon \cdot b = \frac{1}{3q(n)}$$

Combining all of the above we find that

$$|\Lambda - \overline{\mathrm{Tr}_b}(H)| \leq |\Lambda - \Gamma| + |\Gamma - \overline{\mathrm{Tr}_b}(H)| \leq |\Lambda - \Gamma| + |\mathcal{B} - \overline{\mathrm{Tr}_b}(H)| + |\mathcal{E}| \leq \frac{1}{3q(n)} + \Delta + \Delta = \frac{1}{q(n)}.$$

□

II. QUANTUM ALGORITHMS FOR SUES AND LLSD

In this section we give a quantum algorithm for SUES and a quantum algorithm for LLSD. Moreover, if the input is a log-local Hamiltonian, then the quantum algorithms we give in this section turn out to be a $\mathsf{DQC1}$ algorithm in the case of LLSD, and a $\mathsf{DQC1}_{\log n}$ algorithm in the case of SUES. That is, if the input is a log-local Hamiltonian, then these algorithms can be implemented in the one clean qubit model, where in the case of SUES we need to measure logarithmically many qubits (as opposed to just one), in order to read out the entire encoding of the eigenvalue.

By scaling the input $H' = H/\Lambda$, where $\Lambda \in \mathcal{O}(\text{poly}(n))$ is an upper bound on the largest eigenvalue of H, we can assume without loss of generality that $||H|| < 1$. Moreover, we will use that allowing up to $\mathcal{O}(\log(n))$ clean qubits does not change the class $\mathsf{DQC1}$ [1]. That is, the class of problems that can be solved in polynomial time using the one clean qubit model of computation is the same as the class of problems that can be solved in polynomial time using the k-clean qubit model of computation, for $k \in \mathcal{O}(\log n)$. We use this result since the quantum algorithms we describe need additional ancilla qubits, which have to be initialized in the all-zeros state and hence be 'clean'.

A. Quantum algorithm for SUES

In this section we describe a quantum algorithm for SUES, which when the input is restricted to log-local Hamiltonians turns out to be a $\mathsf{DQC1}_{\log n}$ algorithm. That is, if the input is a log-local Hamiltonian, then this algorithm can be implemented using the one clean qubit model of computation where we are allowed to measure logarithmically many of the qubits at the end, in order to read out the encoding of the eigenvalue.

The quantum algorithm for SUES implements an approximation of the unitary e^{iH} using Hamiltonian simulation, to which it applies quantum phase estimation with the eigenvector register starting out in the maximally mixed state. In the remainder of this section we will show that we can control the errors such that quantum phase estimation applied to the approximation of e^{iH} outputs the corresponding eigenvalue of H up to precision $\delta \in \Omega(1/\text{poly}(n))$, with error probability $\mu \in \Omega(1/\text{poly}(n))$. Because the maximally mixed state is in a given eigenstate with uniform probabilities over all eigenstates, this shows that this quantum algorithm is able to output a sample from a (δ, μ)-approximation of the uniform distribution over the eigenvalues of H.

Errors can arise in two places, namely due to the imprecisions of the unitary implemented by the Hamiltonian simulation and due to the imprecisions of estimating eigenvalues using quantum phase estimation. First, we discuss the errors of the Hamiltonian simulation step. Given sparse access to H, we can implement a unitary V such that

$$||V - e^{iH}|| < \gamma, \tag{5}$$

in time $\mathcal{O}(\text{poly}(n, \log(1/\gamma)))$ [3]. The algorithms for Hamiltonian simulation of matrices specified by an oracle unfortunately require more than $\mathcal{O}(\log n)$ ancilla qubits, which implies that they can not be implemented using the one clean qubit model. On the other hand, if H is a log-local Hamiltonian, then Hamiltonian simulation techniques based on the Trotter-Suzuki formula can implement a unitary V that satisfies Equation 5 in time $\mathcal{O}(\text{poly}(n, 1/\gamma))$ [4], while only using a constant number of ancilla qubits [5]. Therefore, if H is a log-local Hamiltonian, then using the one clean qubit model we can implement a unitary V that satisfies Equation 5 in time $\mathcal{O}(\text{poly}(n, 1/\gamma))$.

Denote by λ_j and ζ_j the output of the quantum phase estimation routine (where for now we assume that it works perfectly, i.e., introduces no error) when run using e^{iH} and V, respectively. Then, by Equation 5 we have

$$|e^{i\lambda_j} - e^{i\zeta_j}| \leq \gamma,$$

where we assume that $|\lambda_j - \zeta_j| \leq \pi$ by adding multiples of 2π to λ_j if necessary. With some algebra [5], we can show that this implies that

$$|\lambda_j - \zeta_j| \leq \pi\gamma/2.$$

Choosing the accuracy of the Hamiltonian simulation to be $\gamma = \delta/\pi \in \Omega(1/\text{poly}(n))$, we get that

$$|\lambda_j - \zeta_j| < \delta/2. \tag{6}$$

Next, we will consider the errors that arise from using the quantum phase estimation routine to estimate the eigenvalues ζ_j of the unitary V. The quantum phase estimation routine requires a register of t ancilla qubits (also called the eigenvalue register), onto which the eigenvalue will be loaded. If we take

$$t = \log(2/\delta) + \lceil \log(2 + 1/2\mu) \rceil \in \mathcal{O}(\log n)$$

qubits in the eigenvalue register, then quantum phase estimation outputs an estimate $\overline{\zeta_j}$ that satisfies

$$|\overline{\zeta_j} - \zeta_j| \leq \delta/2,$$

with probability at least $(1-\mu)$ [6]. In particular, with probability at least $(1-\mu)$ this estimate satisfies

$$|\overline{\zeta_j} - \lambda_j| \leq |\overline{\zeta_j} - \zeta| + |\zeta_j - \lambda_j| \leq \delta.$$

This requires $\widetilde{\mathcal{O}}(2^t) = \widetilde{\mathcal{O}}(\mathrm{poly}(n))$ applications of the unitary V, each of which can be implemented in $\mathcal{O}(\mathrm{poly}(n))$ time as discussed above. In addition, this quantum phase estimation step requires only $\mathcal{O}(\log n)$ ancilla qubits, making it possible to be implemented using the one clean qubit model.

In conclusion, both the Hamiltonian simulation and the quantum phase estimation can be implemented up to the required precision in time $\mathcal{O}(\mathrm{poly}(n))$. Moreover, if H is a log-local Hamiltonian, then this can be done using the one clean qubit model. Finally, to read out the encoding of the eigenvalue, we need to measure the $t \in \mathcal{O}(\log(n))$ qubits in the eigenvalue register, resulting in a $\mathsf{DQC1}_{\log n}$ algorithm for SUES if the input is a log-local Hamiltonian.

B. Quantum algorithm for LLSD

In this section, we will describe two quantum algorithms for LLSD, both of which turn into $\mathsf{DQC1}$ algorithms when the input is restricted to log-local Hamiltonians. That is, if the input is a log-local Hamiltonian, then these algorithms can be implemented using the one clean qubit model of computation.

1. *Counting eigenvalues below the threshold*

A straightforward approach is to solving LLSD is to repeatedly sample from the output of SUES and then compute the fraction of samples that lie below the given threshold. The downside of this is that it requires one to measure the entire eigenvalue register consisting of logarithmically many qubits, which is prohibitive as we are only allowed to measure a single qubit in the one clean qubit model. This can be circumvented by simply adding an extra clean qubit and flipping this qubit conditioned on the state in the eigenvalue register being smaller than the given threshold. This extra qubit will be flipped with probability close to the low-lying spectral density, allowing us to obtain a solution to LLSD by only measuring this single qubit. Moreover, if H is a log-local Hamiltonian, then this 'fully quantum' algorithm can be implemented using the one clean qubit model, as it requires only a few more additional clean qubits on top of those required for the quantum algorithm for SUES discussed in Section II A.

Note that the outcome probabilities of this 'fully quantum' algorithm are identical to those obtained by measuring the entire eigenvalue register, followed by classical counting of the number of samples below the given threshold. Consequently, the same error analysis applies in both cases. In the rest of this section we will discuss the error analysis of classically counting the number of samples below the given threshold.

Let $m = \epsilon^{-2} \in \mathcal{O}(\mathrm{poly}(n))$ and draw for $j = 1, \ldots, m$ a sample $\overline{\lambda}_{k_j}$ from SUES with $\delta/2$ as the precision parameter. Next, compute

$$\chi_j = \begin{cases} 1 \text{ if } \overline{\lambda}_{k_j} \in (a - \delta/2, b + \delta/2), \\ 0 \text{ otherwise.} \end{cases}$$

For now we assume that all samples $\overline{\lambda}_{k_j}$ were *correctly sampled*, i.e., each k_j is drawn uniformly at random from the set $\{0, \ldots, 2^n - 1\}$ and $|\lambda_{k_j} - \overline{\lambda}_{k_j}| \leq \delta/2$, where λ_{k_j} denotes the eigenvalue of which $\overline{\lambda}_{k_j}$ is an estimate. We now show that under this assumption the quantity

$$\chi := \frac{1}{m} \sum_{j=1}^{m} \chi_j \tag{7}$$

is, with high probability, a correct solution to LLSD. By the Chernoff-Hoeffding inequality χ is, with high probability, an estimate to within additive precision ϵ of

$$y := \Pr_{\overline{\lambda} \sim \text{SUES}} \left[\overline{\lambda} \in (a - \delta/2, b + \delta/2) \right],$$

where the probability is taken over the $\overline{\lambda}$ being correctly sampled from SUES. Because we assume that the $\overline{\lambda}$ are correctly samples from SUES, we know that they satisfy $|\lambda - \overline{\lambda}| \leq \delta/2$, where λ denotes the eigenvalue of which $\overline{\lambda}$ is an estimate. This implies that

(i) $y \leq \Pr_{\lambda \sim_U \{\lambda_j\}_{j=1}^{2^n}} \Big[\lambda \in (a-\delta, b+\delta)\Big] = N_H(a-\delta, b+\delta)$,

(ii) $y \geq \Pr_{\lambda \sim_U \{\lambda_j\}_{j=1}^{2^n}} \Big[\lambda \in (a, b)\Big] = N_H(a, b)$,

where the probabilities are taken over the λ being sampled uniformly from the set of all eigenvalues of H. Combining this with the Chernoff-Hoeffding inequality, we find that χ is, with high probability, an estimate of y up to additive precision ϵ, where y satisfies

$$N_H(a,b) \leq y \leq N_H(a-\delta, b+\delta).$$

That is, if all $\overline{\lambda}_{k_j}$ were sampled correctly from SUES, then χ is with high probability a correct solution to LLSD.

Finally, we consider the probability that all samples $\overline{\lambda}_{k_j}$ were indeed sampled correctly. By the union bound this probability is at least $1 - m\mu$, where μ denotes the sampling error probability of SUES. Because $m \in \mathcal{O}(\text{poly}(n))$, we can choose $\mu \in \Omega\left(1/\text{poly}(\epsilon^{-2}, n)\right) = \Omega\left(1/\text{poly}(n)\right)$ such that all our samples are sampled correctly with probability close to 1. Therefore, we conclude that the χ defined in Equation 7 is a correct solution to LLSD, with probability close to 1. Moreover, χ can be obtained from a polynomial number of samples from SUES, and can therefore be computed in time $\mathcal{O}(\text{poly}(n))$.

2. *Using trace estimation of eigenvalue transform*

In our paper, we use a result of Cade & Montanaro [5] to argue that the complexity of estimating the spectral entropy of a Hermitian matrix is closely related to DQC1. In their work, Cade & Montanaro describe a DQC1 algorithm can estimate traces of general functions of Hermitian matrices (i.e., beyond spectral entropies). This algorithm could also be used to extract other interesting properties encoded in the spectrum of the combinatorial Laplacian. To illustrate this and connect even further to this line of work, we provide an alternative algorithm for LLSD based on this algorithm. The main result we will utilize is the following Lemma.

Lemma 3 (Cade & Montanaro [5]). *For a log-local Hamiltonian $H \in \mathbb{C}^{2^n \times 2^n}$, and any log-space polynomial-time computable function $f : I \to [-1,1]$ (where I contains the spectrum of H) that is Lipschitz continuous with constant K (i.e., $|f(x) - f(y)| \leq K|x-y|$ for all $x, y \in I$), there exists a DQC1 algorithm to estimate $\mathrm{Tr}(f(H))/2^n = \sum_j f(\lambda_j)/2^n$ up to additive accuracy $\epsilon(K+1)$, where λ_j denote the eigenvalues of H, and $\epsilon \in \Omega(1/poly(n))$.*

It is clear that if the function f is the step-function with threshold $b + \delta/2$ given by

$$f(x) = \begin{cases} 1 \text{ if } x \leq b + \delta/2, \\ 0 \text{ otherwise,} \end{cases}$$

then the quantity estimated by the algorithm of Lemma 3 is a correct solution to LLSD. However, as this function is not Lipschitz continuous, we will use a smooth approximation based on the following lemma.

Lemma 4 (Smooth approximation of the sign function). *Let $\delta > 0$, $\epsilon \in (0,1)$ and $\gamma = \frac{\delta\sqrt{2\epsilon - \epsilon^2}}{1-\epsilon}$. Then, the function $g_\gamma(x) = \frac{x}{\sqrt{x^2+\gamma^2}}$ satisfies*

(i) for all $x \in [-2,2]$: $-1 \leq g_\gamma(x) \leq 1$,

(ii) for all $x \in [-2,2] \backslash (-\delta, \delta)$: $|g_\gamma(x) - sgn(x)| \leq \epsilon$, and

(iii) $\sup_{x \in [-2,2]} |g'_\gamma(x)| \leq \frac{1}{\gamma}$.

Proof. (*i*) It is clear that for all $x \in [-2,2]$ we have: $-1 \leq g_\gamma(-2) \leq g_\gamma(x) \leq g_\gamma(2) \leq 1$.

(*ii*) Let $x \in (\delta, 2]$, then

$$|g_\gamma(x) - \text{sgn}(x)| = |g_\gamma(x) - 1| \leq |g_\gamma(\delta) - 1| = \epsilon.$$

For $x \in [-2, -\delta)$ we note that

$$|g_\gamma(x) - \text{sgn}(x)| = |g_\gamma(x) + 1| \leq |g_\gamma(-\delta) + 1| = |-(g_\gamma(\delta) - 1)| = \epsilon.$$

(iii) It is clear that: $\sup_{x\in[-2,2]} |g'_\gamma(x)| = |g'_\gamma(0)| = \frac{1}{\gamma}$.

$\square$

Let $\gamma = \frac{(\delta/2)\sqrt{2\epsilon-\epsilon^2}}{1-\epsilon}$ and define $g = g_\gamma$ as in Lemma 4. We define our smooth approximation of the step-function by

$$\hat{f}(x) = \frac{g(-x+b')+1}{2},$$

where $b' = b + \delta/2$. By Lemma 4 we know that $\hat{f}$ is Lipschitz continuous on $[0,1]$ with constant $1/\gamma \in \mathcal{O}(\text{poly}(n))$, and that it satisfies

- for all $x \in [0,1] :\ 0 \leq \hat{f}(x) \leq 1$, and
- for all $x \in [0,1]\backslash(b, b+\delta) :\ |\hat{f}(x) - f(x)| \leq \epsilon/2$.

Subsequently, we define our estimation objective

$$y = \frac{1}{2^n}\left(\sum_{j\,:\,\lambda_j\in[0,b]} f(\lambda_j) + \sum_{j\,:\,\lambda_j\in[b,b+\delta]} \hat{f}(\lambda_j)\right),$$

and we note that y indeed satisfies $N_H(0,b) \leq y \leq N_H(0,b+\delta)$, since

$$\begin{aligned} y &= \frac{1}{2^n}\left(\sum_{j\,:\,\lambda_j\in[0,b]} f(\lambda_j) + \sum_{j\,:\,\lambda_j\in[b,b+\delta]} \hat{f}(\lambda_j)\right) \\ &= N_H(0,b) + \underbrace{\frac{1}{2^n}\sum_{j\,:\,\lambda_j\in[b,b+\delta]} \underbrace{\hat{f}(\lambda_j)}_{\in[0,1]}}_{\in[0,N_H(b,b+\delta)]}. \end{aligned}$$

Now our goal is to use Lemma 3 to obtain an ϵ-approximation of y. To this end, we first define

$$\Lambda = \frac{1}{2^n}\sum_{j=1}^{2^n} \hat{f}(\lambda_j),$$

and we upper-bound the absolute difference between y and Λ as follows

$$\begin{aligned} \Big|\Lambda - y\Big| &\leq \frac{1}{2^n}\left|\sum_{j\,:\,\lambda_j\in[0,b]} \Big(\hat{f}(\lambda_j) - f(\lambda_j)\Big) + \sum_{j\,:\,\lambda_j\in[b+\delta,1]} \hat{f}(\lambda_j)\right| \\ &\leq \frac{1}{2^n}\left(\sum_{j\,:\,\lambda_j\in[0,b]} \Big|\hat{f}(\lambda_j) - f(\lambda_j)\Big| + \sum_{j\,:\,\lambda_j\in[b+\delta,1]} \Big|\hat{f}(\lambda_j)\Big|\right) \\ &\leq \frac{1}{2^n}\left(\sum_{j\,:\,\lambda_j\in[0,b]} \epsilon/2 + \sum_{j\,:\,\lambda_j\in[b+\delta,1]} \epsilon/2\right) \leq \epsilon/2. \end{aligned}$$

Finally, let χ be the output of the algorithm of Lemma 3 applied to our function $\hat{f}$ with precision parameter $\hat{\epsilon} = \epsilon/(2(K+1)) \in \Omega(1/\text{poly}(n))$. In particular, χ satisfies $|\chi - \Lambda| \leq \hat{\epsilon}(K+1) = \epsilon/2$. We conclude that χ is a correct solution to LLSD since

$$|\chi - y| \leq |\chi - \Lambda| + |\Lambda - y| \leq \epsilon/2 + \epsilon/2 = \epsilon,$$

and y indeed satisfies $N_H(0,b) \leq y \leq N_H(0,b+\delta)$ as discussed earlier.

III. SWES IS DQC1-HARD

In this section, we will show that SWES is DQC1-hard. We will do so by showing that we estimate the DQC1-hard normalized subtrace $\overline{\mathrm{Tr}}_b(H)$ from Proposition 1 up to additive polynomial precision $\epsilon \in \Omega(1/\mathrm{poly})$ using a polynomial number of queries to an oracle for SWES, together with polynomial-time classical preprocessing of the input and postprocessing of the output.

First, by considering how H is constructed in [2], we note that $\mathrm{Tr}(H)$ is known and that $\mathrm{Tr}(H)/2^n \in \mathcal{O}\left(\mathrm{poly}(n)\right)$. Next, we define $\hat{\epsilon} = (\epsilon/(\mathrm{Tr}(H)/2^n))$ and $m = 1/\hat{\epsilon}^2$. Subsequently, let $\overline{\lambda}_{k_1}, \ldots, \overline{\lambda}_{k_m}$ denote samples drawn from SWES with estimation precision $\delta/2$, where δ is such that H has no eigenvalues in $[b, b+\delta]$. For now we assume that all samples were *correctly sampled*, i.e., $|\overline{\lambda}_{k_j} - \lambda_{k_j}| \le \delta/2$, where λ_{k_j} denotes the eigenvalue of which $\overline{\lambda}_{k_j}$ is an estimate. Afterwards, we estimate the normalized subtrace $\overline{\mathrm{Tr}}_b(H)$ by computing the ratio of samples that is below $b + \delta/2$

$$\chi = \frac{1}{m} \sum_{j\,:\,\overline{\lambda}_{k_j} \le b+\delta/2} 1$$

By the Chernoff-Hoeffding inequality (together with the fact that H has no eigenvalues in $[b, b+\delta]$), this ratio χ is, with high probability, an estimate of

$$\Lambda = \sum_{j\,:\,\lambda_j \le b} \lambda_j/\mathrm{Tr}(H),$$

up to additive precision $\hat{\epsilon}$. Therefore, $(\mathrm{Tr}(H)/2^n) \cdot \chi$ is, with high probability, an ϵ estimate of $(\mathrm{Tr}(H)/2^n) \cdot \Lambda = \overline{\mathrm{Tr}}_b(H)$.

Finally, we consider the probability that all samples $\overline{\lambda}_{k_j}$ were indeed sampled correctly. By the union bound this probability is $M \cdot \mu$, where μ denotes the sampling error probability of SWES. Because $m \in \mathcal{O}\left(\mathrm{poly}(n)\right)$, we can choose $\mu \in \Omega(1/m) = \mathcal{O}\left(\mathrm{poly}(n)\right)$ such that all our samples are sampled correctly with probability close to 1.

[1] P. W. Shor and S. P. Jordan, Quantum Information & Computation **8**, 681 (2008), `arXiv:0707.2831`.

[2] F. G. Brandão, *Entanglement theory and the quantum simulation of many-body physics*, Ph.D. thesis, University of London (2008), `arXiv:0810.0026`.

[3] G. H. Low and I. L. Chuang, Physical review letters **118**, 010501 (2017), `arXiv:1606.02685`.

[4] S. Lloyd, Science , 1073 (1996).

[5] C. Cade and A. Montanaro, in *13th Conference on the Theory of Quantum Computation, Communication and Cryptography* (2018) `arXiv:1706.09279`.

[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. (Cambridge University Press, 2011).