Quantifying Quantum Advantage in Topological Data Analysis

Dominic W. Berry,[1,*] Yuan Su,[2] Casper Gyurik,[3] Robbie King,[2,4] Joao Basso,[2] Alexander Del Toro Barba,[2] Abhishek Rajput,[5] Nathan Wiebe,[5,6] Vedran Dunjko,[3] and Ryan Babbush[2,†]

[1] *Department of Physics and Astronomy, Macquarie University, Sydney, NSW 2109, Australia*
[2] *Google Quantum AI, Venice, CA 90291, United States*
[3] *applied Quantum algorithms (aQa), Leiden University, 2300 RA Leiden, The Netherlands*
[4] *Department of Computing and Mathematical Sciences, Caltech, Pasadena, CA 91125, United States*
[5] *Department of Computer Science, University of Toronto, ON M5S 2E4, Canada*
[6] *Pacific Northwest National Laboratory, Richland, WA 99354, United States*

(Dated: September 28, 2022)

Lloyd *et al.* [1] were first to demonstrate the promise of quantum algorithms for computing Betti numbers in persistent homology (a way of characterizing topological features of data sets). Here, we propose, analyze, and optimize an improved quantum algorithm for topological data analysis (TDA) with reduced scaling, including a method for preparing Dicke states based on inequality testing, a more efficient amplitude estimation algorithm using Kaiser windows, and an optimal implementation of eigenvalue projectors based on Chebyshev polynomials. We compile our approach to a fault-tolerant gate set and estimate constant factors in the Toffoli complexity. Relative to the best classical heuristic algorithms, our analysis reveals that super-quadratic quantum speedups are only possible for this problem when targeting a multiplicative error approximation and the Betti number grows asymptotically. Further, we propose a dequantization of the quantum TDA algorithm that shows that having exponentially large dimension and Betti number are necessary, but insufficient conditions, for super-polynomial advantage. We then introduce and analyze specific problem examples for which super-polynomial advantages may be achieved, and argue that quantum circuits with tens of billions of Toffoli gates can solve some seemingly classically intractable instances.

I. INTRODUCTION

An important outstanding challenge in quantum computing is to find quantum algorithms that provide a significant speedup for practical problems. One area of great interest is quantum machine learning [2]. Early proposals included, for example, principal component analysis [3], and were often based on quantum solution of linear equations [4]. However, it has proven possible to dequantize many of these proposals, indicating that there is at most a polynomial speedup [5, 6]. Analysis of the cost taking into account error-correction overhead indicates that more than a quadratic speedup would be needed to provide a useful quantum advantage within quantum error-correction [7, 8].

An algorithm for topological data analysis proposed by Lloyd *et al.* [1] turned out not to be directly "dequantizable" using the same techniques, raising the question of whether a greater speedup was possible. A simple analysis by Gunn *et al.* [9] contradicted some of the scaling results originally reported by Lloyd *et al.*, and indicated that under certain assumptions there would still only be a quadratic speedup for these algorithms (our analysis agrees with that of Gunn *et al.*). Here we give a far more careful analysis of the complexity, and examine applications which provide better-than-quadratic speedups.

An important goal in data analysis is to extract features of a data set and use them to cluster or classify the data. This data set would be represented as a set of points in some metric space, such as $\mathbb{R}^n$ with the Euclidean distance function. One approach for the analysis is to convert the point cloud into a graph where the vertices are the given data points and the edges are determined by whether or not pairs of points lie within a chosen distance ϵ. This approach can capture features such as connectivity but ignores potential higher dimensional features, especially if the data points are sampled from some underlying high-dimensional manifold. Topological data analysis (TDA) attempts to extract such higher dimensional global topological features of an underlying data set by applying techniques from the field of algebraic topology, in particular what is known as simplicial homology.

* corresponding author: dominic.berry@mq.edu.au
† corresponding author: ryanbabbush@gmail.com

A simplex is a point, line segment, triangle, or higher-dimensional equivalent, and a simplicial complex is a collection of simplexes. One can form a simplicial complex from the data set with respect to a distance scale ϵ, by adding points that are within distance 2ϵ to simplices. The Betti number β_k is the number of k-dimensional holes of the complex. One can determine the Betti number for a chosen range of ϵ. Betti numbers which persist over an appreciable range of the values of ϵ are indicative of intrinsic topological features of the data set, as opposed to artifacts that appear at a particular scale and disappear shortly thereafter. The study of such features is referred to as persistent homology.

The classical complexities of algorithms for estimating Betti numbers are typically exponential in k. That means the computation can be intractable even for a moderate amount of data. That is an important feature for the promise of quantum algorithms, because even fully error-corrected quantum computers with millions of physical qubits are expected to be very limited in data storage. The most promising applications of quantum computers are therefore those involving a limited amount of classical data that needs to be fed into the quantum algorithm as part of the problem specification.

Recent work on quantum TDA algorithms [10, 11] introduced more efficient fermionic representations of the Dirac operator and employed the quantum singular value transformation to implement the kernel projector. Some of these techniques have led to significant asymptotic improvements over the original approach, but it is unclear whether they are useful for reducing the fault-tolerant implementation cost for solving problems in practice. Indeed, to the best of our knowledge, no study has been done on the fault-tolerant implementation of quantum TDA algorithms for solving *any* instance of problems of practical interest.

In this work, we give a new algorithm for estimating Betti numbers on a quantum computer. We provide a number of improvements which not only streamline the algorithmic analyses but also reduce the cost of fault-tolerant implementation. Specifically, we develop a new method to prepare the initial Dicke states, introduce improved amplitude estimation using Kaiser windows, directly construct the quantum walk operator from block encoding, optimally project onto the kernel of the boundary map, then use the overlap estimation to estimate the kernel dimension of the block-encoded operator, leading to a quadratic improvement in precision over classical sampling. We also provide the concrete constant factors in the complexity of our algorithm and estimate its fault-tolerant cost, going beyond the asymptotic analyses of all existing work on quantum topological data analysis. Finally, we show that it is possible to construct specific data sets for which quantum TDA would appear to have a significant speedup. In particular, we give examples of a very specific problem where super-polynomial speedup is possible and a more general problem where quartic speedup is possible. Here, we are comparing to well studied heuristic classical approaches with complexity approximately linear in the possible number of cliques. Formally exact classical algorithms also exist for this problem but scale at least quadratically in the number of cliques.

We provide a more detailed explanation of the technical background needed to understand Betti numbers in Section II. We then provide the improved algorithm and the analysis of its complexity in Section III. We use this result to analyse the regimes where large quantum speedups may be expected in Section IV. In particular, we consider cases where the Betti number would be large (implying a large quantum speedup) in Section IV A, and novel competing classical algorithms in Section IV D. We then conclude in Section V.

II. TECHNICAL BACKGROUND

Here we give the more detailed background that is needed to understand the standard approaches for this problem and our contribution. For the technical definitions of the simplicial complex and Betti number, see Appendix A.

A. Overview of the TDA algorithm and its implementation

In order to analyse the Betti numbers, the points and lines between the points are represented by a graph G. Then a simplex is represented by a clique in the graph (groups of vertices that are all connected by edges). The n vertices of the graph G are represented by n qubits. That is, $|1\rangle|0\rangle\cdots|0\rangle$ would represent the first vertex, and $|0\rangle|0\rangle\cdots|1\rangle|0\rangle$ would represent vertex $n-1$. Note that this is a distinct representation from that often used to analyse sparse Hamiltonians, where each computational basis state represents a distinct vertex (so n qubits would represent 2^n vertices). In the representation here, a computational basis state

with more than one $|1\rangle$ would represent a clique of the graph (a set of vertices with edges connecting every pair of vertices). For example, $|1\rangle|1\rangle|1\rangle|0\rangle\cdots|0\rangle$ would represent a clique of the first three vertices.

The entire Hilbert space can then be subdivided into subspaces of different Hamming weights. Using $\mathcal{H}_k$ to denote the space spanned by computational basis states with Hamming weight k, we have

$$\left(\mathbb{C}^2\right)^{\otimes n} = \bigoplus_{k=0}^{n} \mathcal{H}_k, \tag{1}$$

where $\dim(\mathcal{H}_k) = \binom{n}{k}$. This space includes all states of the various Hamming weights. One can also restrict to only states which represent vertices of cliques of the graph G. Denoting by $\mathcal{H}_k^G$ the space spanned by basis states of all k-cliques of G, we have $\mathcal{H}_k^G \subseteq \mathcal{H}_k$. We also use $\mathrm{Cl}_k(G)$ to denote the set of bit strings which correspond to k-cliques of G.

We define boundary maps $\partial_k : \mathcal{H}_{k+1} \to \mathcal{H}_k$ by their actions on the basis states $|x\rangle \in \mathcal{H}_{k+1}$ as

$$\partial_k |x\rangle := \sum_{i=0}^{k} (-1)^i |x\backslash(i)\rangle, \tag{2}$$

where $x\backslash(i)$ means the ith 1 in the bit string x is set to 0. We also define $\partial_k^G : \mathcal{H}_{k+1}^G \to \mathcal{H}_k^G$ as the restriction of ∂_k to $\mathcal{H}_{k+1}^G$. By definition, we have that both $\mathrm{im}(\partial_{k+1}^G)$ and $\ker(\partial_k^G)$ are subspaces of $\mathcal{H}_k^G$. But in fact, we have $\mathrm{im}(\partial_{k+1}^G) \subseteq \ker(\partial_k^G) \subseteq \mathcal{H}_k^G$, which can be seen from (with $|x\rangle \in \mathcal{H}_k^G$)

$$\begin{aligned}
\partial_k^G \partial_{k+1}^G |x\rangle &= \sum_{i=0}^{k+1} (-1)^i \partial_k^G |x\backslash(i)\rangle \\
&= \sum_{i=0}^{k+1} (-1)^i \sum_{j=0}^{i-1} (-1)^j |x\backslash(j,i)\rangle + \sum_{i=0}^{k+1} (-1)^i \sum_{j=i+1}^{k} (-1)^{j-1} |x\backslash(i,j)\rangle \\
&= \sum_{i=0}^{k+1} (-1)^i \sum_{j=0}^{i-1} (-1)^j |x\backslash(j,i)\rangle + \sum_{j=0}^{k+1} (-1)^{j-1} \sum_{i=0}^{j-1} (-1)^i |x\backslash(i,j)\rangle \\
&= 0.
\end{aligned} \tag{3}$$

Since $\mathrm{im}(\partial_{k+1}^G)$ is a subspace of $\ker(\partial_k^G)$, one can define the quotient space

$$H_k(G) := \ker(\partial_k^G)/\mathrm{im}(\partial_{k+1}^G). \tag{4}$$

This space is called the kth homology group, and its dimension

$$\beta_k^G := \dim(H_k(G)) = \dim(\ker(\partial_k^G)) - \dim(\mathrm{im}(\partial_{k+1}^G)) \tag{5}$$

is the kth Betti number. In practice, Betti numbers β_k^G can be used to extract features of the shape of the data modeled by the graph G, and their estimation is the main problem in the topological data analysis we will consider here. In this work we will be estimating β_{k-1}^G for the $k-1$th Betti number, so we can simplify our discussion by considering Hamming weight k.

To describe our quantum algorithm and its circuit implementation for estimating Betti numbers, we will introduce the Dirac operator B_G. Specifically, for any graph G and a fixed value of k, we define

$$B_G := \begin{bmatrix} 0 & \partial_{k-1}^G & 0 \\ \partial_{k-1}^{G\dagger} & 0 & \partial_k^G \\ 0 & \partial_k^{G\dagger} & 0 \end{bmatrix}, \tag{6}$$

where the blocks indicate the subspaces $\mathcal{H}_{k-1}^G$, $\mathcal{H}_k^G$ and $\mathcal{H}_{k+1}^G$. Since $\partial_{k-1}^G \partial_k^G$ gives zero, squaring B_G yields

$$B_G^2 = \begin{bmatrix} \partial_{k-1}^G \partial_{k-1}^{G\dagger} & 0 & 0 \\ 0 & \partial_{k-1}^{G\dagger} \partial_{k-1}^G + \partial_k^G \partial_k^{G\dagger} & 0 \\ 0 & 0 & \partial_k^{G\dagger} \partial_k^G \end{bmatrix}. \tag{7}$$

It can be seen here that the middle part corresponds to the combinatorial Laplacian [12]

$$\Delta_{k-1}^{G} = \partial_{k-1}^{G\dagger}\partial_{k-1}^{G} + \partial_{k}^{G}\partial_{k}^{G\dagger}. \tag{8}$$

It is known that [12–14]

$$\dim(\ker(\Delta_{k-1}^{G})) = \beta_{k-1}^{G}, \tag{9}$$

which provides a convenient way of computing Betti numbers. Since B_G is Hermitian, the kernel of B_G and B_G^2 is identical. Therefore, to estimate the Betti number corresponding to a particular graph and a fixed value of k, it suffices to construct the Dirac operator and compute the dimension of its kernel on the subspace $\mathcal{H}_k^G$.

It can be difficult in general to perform topological data analysis on a classical computer due to the high-dimensional nature of the problem, with the dimension increasing exponentially in k. However, the Dirac operator could be efficiently simulated on a quantum computer, in the sense of solving the Schrödinger equation with the Dirac operator as the Hamiltonian. That indicates exponential speedups are possible, though there are a number of other stages needed for the quantum algorithm. Previous work has provided several approaches for estimating Betti numbers on quantum computers. The stages of these approaches include preparation of a uniformly mixed state, construction of the projector onto the kernel subspace, and estimation of the overlap.

The original approach of Lloyd et al. [1] applied amplitude amplification and estimation to prepare the desired initial state in $\mathcal{H}_k^G$, starting from a uniform mixture of all Hamming weight k basis states in $\mathcal{H}_k$. Their approach actually produces a superposition over all values of k, so the success probability of obtaining a specific k can be quite low; this issue was addressed by later work such as [9, 14]. To construct the projector onto the kernel, they implement Hamiltonian simulation and perform quantum phase estimation on the resulting operator. The Betti number is then estimated as the frequency of zero eigenvalues in the measurement. That is, the algorithm can be summarised as below.

1. For $i = 1, \ldots, m$, repeat:
   (a) Prepare the mixed state
   $$\rho_k^G = \frac{1}{\dim(\mathcal{H}_k^G)} \sum_{x \in \mathrm{Cl}_k(G)} |x\rangle\langle x| . \tag{10}$$
   (b) Apply quantum phase estimation to the unitary $e^{iB_G t}$.
   (c) Measure the eigenvalue register to obtain an approximation $\widetilde{\lambda}_i$.
2. Output the frequency of zero eigenvalues:
   $$\frac{\#\{i, \widetilde{\lambda}_i = 0\}}{m}. \tag{11}$$

In this work, we give a new algorithm for estimating Betti numbers on a quantum computer. We provide a number of improvements which not only streamline the algorithmic analyses but also reduce the cost of fault-tolerant implementation. Specifically, we do the following.

- Develop new methods to prepare a mixture of fixed Hamming-weight states with garbage information that have significantly lower fault-tolerant cost.
- Introduce improved amplitude estimation using Kaiser windows to estimate the number of steps of amplitude estimation needed.
- Directly construct the quantum walk operator from block encoding without an additional step of quantum simulation.
- Project onto the kernel of the boundary map by implementing a Chebyshev polynomial to optimally filter the zero eigenvalues. This is more efficient than previous approaches that implement the phase estimation or rectangular window functions for filtering.

- Use the overlap estimation to estimate the kernel dimension of the block-encoded operator, leading to a quadratic improvement in precision over the classical sampling approach used by previous work.

We also provide the concrete constant factors in the complexity of our algorithm and estimate its fault-tolerant cost for solving example problems, going beyond the asymptotic analyses of all prior work on quantum topological data analysis.

Ultimately, the performance of our algorithm (as well as other algorithms from previous work) will depend on several important problem parameters. First, the desired state on which we perform the kernel projector is a uniform mixture of all the $|\mathrm{Cl}_k(G)|$ basis states in $\mathcal{H}_k^G$, whereas we start with a uniform mixture of all $\binom{n}{k}$ basis states in $\mathcal{H}_k$. Their ratio $\binom{n}{k}/|\mathrm{Cl}_k(G)|$ will determine the number of amplification steps required in the state preparation. There is potential to improve the efficiency of preparation of the cliques via an improved clique-finding algorithm.

Second, we need to implement a spectral projector that distinguishes the zero eigenvalue from the remaining nonzero eigenvalues of the Dirac operator, and the cost of implementing such a projector will depend on the spectral gap of the Dirac operator. Third, the output of the quantum TDA algorithm will not be the actual Betti number but instead a normalized version of the quantity $\beta_{k-1}^G/|\mathrm{Cl}_k(G)|$. In order to estimate the Betti number to some additive precision, we need to increase the complexity by a factor that depends on $|\mathrm{Cl}_k(G)|$, with the result that the complexity would roughly scale as $\sqrt{\binom{n}{k}}$.

An alternative scenario is that a fixed relative error is required; that is, the ratio of the uncertainty in the Betti number to the Betti number. Then the complexity would roughly scale as $\sqrt{\binom{n}{k}/\beta_{k-1}^G}$, as we show in Section III. This means that significant speedups can be provided in cases where the Betti number β_{k-1}^G is large, and we provide examples of such graphs in Section IV.

B. Complexity classes of TDA

Linear-algebraic applications of quantum computing have lead to numerous suggestions of how various types of machine learning subroutines could be implemented on a quantum computer with superpolynomial speed-ups over their classical counterparts. Many of these methods were in the end shown to only suffice for at most polynomial speed-ups, due to the randomized "dequantizations" of Tang and others [5, 6, 15]. The algorithm of Lloyd et al. [1], however, turned out not to be directly "dequantizable" using similar techniques, raising the question of whether more robust complexity-theoretic quantum-classical separations can be proven. The current landscape on this topic is somewhat involved.

In general, we have a number of discrepancies between the computational problems in ordinary TDA applications and the computational problems for which we have certain complexity-theoretic insights. In ordinary TDA applications one is typically concerned with the computation of the exact count of zero eigenvalues of combinatorial Laplacians. By the result of [16] – which shows that deciding if a combinatorial Laplacian has a trivial or non-trivial kernel (i.e., Betti number zero or non-zero) is QMA_1-hard – this problem is likely beyond what is efficient even for quantum computers in the worst case. This observation goes in line with classical bodies of work showing that exact computations of Betti numbers is NP-hard [17], and that it can even be PSPACE-hard for more involved topological spaces (i.e., so-called *algebraic-varieties*) [18].

From the perspective of the types of problems quantum algorithms may be efficient for, one could attempt a few relaxation of the problem. First, it may be fruitful to relax the TDA problem with respect to the quantity estimated. Specifically, instead of the number of exactly zero eigenvalues, one could relax it and count the number of "small" eigenvalues (i.e., below a threshold). This relaxation may be convenient from a quantum algorithmic perspective, but it also still useful from a data analysis perspective, since Cheeger's inequality demonstrates that the magnitudes of the small non-zero eigenvalues of the graph Laplacian characterises the connectedness of the graph [19], and similar results hold for combinatorial Laplacians [20]. In folklore it is conjectured that for difficult cases, the magnitude of the smallest non-zero eigenvalue of combinatorial Laplacians very often scales inverse polynomially [21], in which case the number of "small" eigenvalues coincides with the number of zero eigenvalues if the threshold is chosen appropriately. While the problem of counting small eigenvalues is more suitable to be solved on a quantum computer, it could turn out to still be QMA_1-hard if the TDA matrices have a sufficiently large spectral gap. Specifically, if the TDA matrices are sufficiently gapped, then one could count the number of zero eigenvalues (which is QMA_1-hard [16]) by counting the number of eigenvalues below the spectral gap (i.e., the number of "small" eigenvalues).

A related (yet different) problem for which complexity-theoretical results are known is that of estimating *normalized* Betti numbers to within additive inverse polynomial precision. That is, the number of zero eigenvalues divided by the total number of eigenvalues, which here would be $\beta_{k-1}^G/|\mathrm{Cl}_k(G)|$ (if the TDA matrix is sufficiently gapped). This quantity is natural from a quantum computational complexity perspective (though not from an applications perspective), since a quantum algorithm naturally estimates probabilities (so normalized quantities in this case), and since additive errors allow for a direct relationship to definitions of complexity classes like DQC1.

Specifically, in [14] it was shown that the generalization of this problem, namely that of estimating the ratio when allowing a range of small eigenvalues, rather than strictly zero eigenvalues, for *arbitrary* Hermitian operators (i.e., the so-called *low-lying spectral density*) is DQC1-hard. At present it is unknown whether the hardness persists when only considering combinatorial Laplacians, and the closest result to this is the QMA_1-hardness result of [16] for the problem of exact counting. This normalized quantity is not typically studied, and indeed there are concerns that Betti numbers may fail to be large enough to be detectable when normalized (see also Appendix E).

As discussed above, estimates of the normalized Betti number with additive error are more natural from a quantum computational complexity perspective. However, from the perspective of applications, we typically work with (unnormalized) Betti numbers (and perhaps their estimates). For this case, the rescaling from normalized Betti numbers to Betti numbers causes an in general exponential blow up of additive errors, and leads to algorithms which always have exponential run-times (for constant error). At the same time, in many applications, we only require small additive errors when the quantities in question are themselves small. For these reasons here we focus on estimation to within a given relative error; that is, the error in the Betti number divided by the Betti number. That is immune to rescaling and can lead to efficient algorithms in the cases when the Betti numbers are large.

Note that the problem of estimating the low-lying spectral density up to a certain relative error is also DQC1-hard. The reason is that the relative error must always be at least as large as the error in the normalised quantity, and estimating the normalised quantity to additive precision ϵ is DQC1-hard for $\epsilon = 1/\mathrm{poly}(n)$ [14]. It is unknown whether the hardness result holds for Betti numbers, because they are found by restricting to combinatorial Laplacians, rather than arbitrary Hermitian operators. Generally, the larger the Betti number the more efficient the quantum algorithm will be, which in certain cases results in a polynomial quantum runtime. Examples of cases where the Betti numbers are large are discussed in more detail in Section IV.

III. OPTIMIZATION AND ANALYSIS OF QUANTUM TOPOLOGICAL DATA ANALYSIS

A. Generating Dicke states with garbage

In this section, we consider preparing an n-qubit uniform superposition of Hamming weight k basis states (which is allowed to be entangled with garbage states). Such a state is known in previous literature as the Dicke state. In [22] it was shown how to prepare a Dicke state with $\mathcal{O}(nk)$ gates, although these gates included rotations, so there would be a logarithmic factor in the complexity when counting non-Clifford gates.

Because the preparation here allows an entangled state to be prepared between the superposition for the Dicke state and ancilla states, it is possible to prepare the state more efficiently. One approach is to apply a quantum sort to n registers, then use it to apply an inverse sort to the n qubits with the first k set in the state $|1\rangle$. This is similar to the approach used for symmetrising states for chemistry in [23]. Another approach is to use inequality testing to obtain k successes. Both those approaches give a factor of $\log n$ in the number of qubits required, which is costly when n is large.

It is also possible to give a more ancilla-efficient preparation scheme; see Appendix B. That scheme is approximately as Toffoli-efficient when $\binom{n}{k} \sim n^k/k!$. However, for the cases we will consider below with a large quantum speedup, there is a very large number of Toffolis required but only a moderate number of qubits. In such a situation, the implementation on a quantum computer in the surface code would be designed with many parallel Toffoli factories to achieve the result in a reasonable time, as in [24]. In the tradeoff between Toffolis and ancilla qubits, it is more effective to use more ancillas so that fewer Toffoli factories are needed, so we present the more Toffoli-efficient scheme here.

In order to prepare the Dicke state in a Toffoli-efficient way, we prepare n registers with approximately $\log n$ qubits in equal superposition. This is similar to the first step in [23] where a sort was used. Here

we instead find a threshold such that k registers are less than or equal to this threshold. The steps of this procedure are as follows.

1. Prepare n registers in an equal superposition of $f(n)$ values, where $f(n)$ is a power of 2 greater than or equal to cn for some constant c. This may be done with Hadamards (no non-Clifford gates).
2. Sum the most significant bits of these registers. This may be done with no more than n Toffolis. This gives the number of registers at least as large as $10000\ldots$.
3. Perform an inequality test with k (with $\log n$ Toffolis) and place the result in an ancilla. That is, we are testing if the number is $\geq k$, and we call the bit representing the result b_1.
4. For $j = 2$ to $n_{\text{seed}} = \lceil \log f(n) \rceil$, we perform the following.
   (a) Perform an inequality test on each register checking if the first j bits are $\geq b_1 \ldots b_{j-1}1$. The cost is $(j-1)n$ Toffolis.
   (b) Sum the results of those inequality tests, with cost n.
   (c) Perform an inequality test if that number is $\geq k$, setting the result as b_j, with cost $\log n$.
5. Perform an inequality test if all n_{seed} bits are $\geq b_1 \ldots b_{n_{\text{seed}}}$. The cost is $n_{\text{seed}} n$ Toffolis.
6. Sum the results of those inequality tests, with cost n.
7. Check if the result is equal to k, with cost $\log n$.

The logic of this procedure is that whenever there are more than k we increase the threshold which is given by the bits $b_1 b_2 \ldots$, otherwise we reduce it. At the end, provided we have success where there sum of the ones was equal to k, then we have k ones in a superposition of permutations corresponding to a Dicke state. These are obtained by the inequality tests in step 5, and we check there are exactly k ones in steps 6 and 7. It is entangled with the ancilla registers, but this is suitable for our application. Summing all the costs in this procedure gives a total Toffoli cost

$$(n_{\text{seed}} + 1)\left[\frac{n}{2}(n_{\text{seed}} + 2) + \lceil \log n \rceil\right]. \tag{12}$$

Numerically, we find that the probability of failure is approximately $1/2c$. To provide an analytic argument for this failure rate, note that the failure occurs where it is not possible to provide a threshold where there are k numbers greater than or equal to it. That will be true if the kth smallest and $k+1$th smallest numbers are equal. In other words, if we were to sort our numbers, and compare the values in register k and register $k+1$, there would be failure if these numbers were equal. (Note that we do not perform this sort in practice.) Considering the number in register k, there will be success if none of the other $n-1$ numbers are equal. Since there are cn possible values, the probability of a single number not being equal is $1 - 1/cn$, and the probability of all $n-1$ not being equal is $(1-1/cn)^{n-1}$.

That would give a probability that at least one is the same value (as register k) as $1-(1-1/cn)^{n-1} \approx (n-1)/cn \approx 1/c$. That would potentially give failure, but it is (approximately) equally likely that the sort will give this equal value before or after register k. That gives a factor of 2 for a failure probability of $1/2c$. In practice, we find that moderate constant values of c are suitable for minimising the complexity, and we will take $c = 8$ in examples below. That only increases the cost about 3%, while only needing another 3 qubits to store the numbers. The result of taking c constant is that the Toffoli complexity is approximately $(n/2)\log^2 n$.

In comparison, in the sorting approach from [23], the complexity is approximately $2n_{\text{seed}} n \log n$, where $n \log n$ is the number of steps in the sorting network. Moreover, that approach fails when *any* two seed numbers have equal values, which requires taking $f(n) \gtrsim n^2$ to provide a reasonable probability of success. That results in a preparation complexity of approximately $2n\log^2 n$, or 4 times what we have provided here.

As a simple example where our threshold procedure is performed, consider $n = 4$ and $c = 4$, so we have 4 registers of 4 qubits. An example of a basis state is

$$|0110\rangle\,|1110\rangle\,|0111\rangle\,|0010\rangle\,. \tag{13}$$

Now say we aim for $k = 2$. The steps are as follows.

- Sum the first (most significant bits) giving 1. This is equivalent to checking how many registers are at least 1000. The sum is 1, which is less than k, so $b_1 = 0$.
- Perform an inequality test on the first two bits of each register with 01. We get a total of 3, which is greater than k, so $b_2 = 1$.
- Perform an inequality test of the first three bits with 011. We get 3 again, so $b_3 = 1$.
- Perform an inequality test with 0111. We get a total of 2, which is equal to k, so we get $b_4 = 1$.
- Perform an inequality test with 0111 again. Now test equality with k, which succeeds, so we get overall success.

As an alternative to the above example, consider the case that the basis state is

$$|0110\rangle\,|1110\rangle\,|0110\rangle\,|0010\rangle\,. \tag{14}$$

Then in the second-last step, the total would be 1, so we would have $b_4 = 0$, and in the last step we would test inequality with 0110. That would give three greater than or equal to 0110, so we would fail the test that the number is equal to k. In this case, if you sort the numbers you have 0010, 0110, 0110, 1110, and you can see that the second and third numbers are equal. This is the condition for failure discussed above.

B. Detecting the cliques

In the previous section, we have discussed the preparation of the n-qubit Dicke state with Hamming weight k (and an additional garbage register)

$$\frac{1}{\sqrt{\binom{n}{k}}}\sum_{|x|=k}|x\rangle\,\frac{1}{\sqrt{k!(n-k)!}}\sum_{\sigma(0\cdots01\cdots1)=x}|\sigma\rangle\,. \tag{15}$$

Here, the first register holds all n-qubit strings with Hamming weight k, representing subsets of k vertices in an n-vertex graph G. We now describe a quantum circuit that detects whether a given string x represents a k-clique in the underlying graph, with the promise that x has Hamming weight k. Specifically, our goal is to implement the mapping

$$|x\rangle\,|0\rangle\,|0\rangle \mapsto |x\rangle\,|x \in \mathrm{Cl}_k(G)\rangle\,|\mathrm{garb}_x\rangle\,. \tag{16}$$

Here, the second register has value 1 if x represents a k-clique in G and 0 otherwise. The third register contains some garbage information garb_x that can depend on x and need not be uncomputed.

Our implementation of the clique detection is related to the approach of [25]. Specifically, we introduce a register of $\lfloor \log \binom{k}{2} \rfloor + 1 \leq 2 \log k$ qubits to represent integers $0, \ldots, \binom{k}{2}$. This register will be used to count the number of edges in the subgraph induced by the k vertices denoted by x. For the graph, we assume that it is given by a classical database, so we need to run through this classical database, rather than assuming any oracular access to the graph. Let us assume that we have a listing of all edges in the graph. That is, for each edge, we have a listing of the two nodes. In order to implement this classical data, for each edge in the list we use a Toffoli with the qubits representing those two nodes as controls, and an ancilla as target. In the case where both qubits are in the state 1, the ancilla qubit will be flipped.

The complexity is then given by a number of Toffolis equal to the number of edges, which we denote $|E|$. We aim to sum all the bits output by these Toffolis. Provided that we are restricted to Hamming weight k, if x represents a k-clique then every pair of ones in x will result in a 1, so the sum will yield $\binom{k}{2}$. Summing bits in the obvious way would yield a complexity scaling as $2|E| \log k$ Toffolis, because each addition requires multiple Toffolis. An improved method is given in [26], where it would take no more than $|E|$ Toffolis, but the same number of ancillas would be required, which would typically be a prohibitively large cost. An alternative way of summing bits is given in [27], where multiple groups of bits are summed, and their sums are summed. The overall complexity is no more than $2|E|$ Toffolis, and only a logarithmic number of ancillas is used. The costs of the three main parts of the algorithm are as follows.

1. There is cost $|E|$ Toffolis for checking the edges of the graph. The resulting qubits can be erased with measurements and phase corrections, with zero Toffoli cost.

2. The complexity of the efficient bit sum approach from [27] is $2|E|$.

3. There is complexity no more than $2\log k$ Toffolis to check that the output register is equal to $\binom{k}{2}$.

Therefore, the total cost of clique detection is no more than $3|E| + 2\log k$ Toffolis. In many cases we will need to reflect on the result of this test. In that case the $2\log k$ cost is not doubled, because we can replace the equality test with a controlled phase. Therefore the cost in that case is $6|E| + 2\log k$. If we were to retain the qubits resulting form the edge checking and use the sum from [26], the cost would be $2|E|+2\log k$, though with a large ancilla cost.

Later when we consider the block encoding of the Hamiltonian we will need to allow a wider range of Hamming weights, $k-1$, k and $k+1$, in the case where we are block encoding the Hamiltonian projected onto this subspace. First we can sum the ones in the string x, which has Toffoli complexity n. We can check if the sum is equal to $k-1$ with $\lceil \log n \rceil$ Toffolis, then check if it is k or $k+1$ with further Toffolis with the unary iteration procedure. The number of Toffolis needed depends on the value of k, and some values will require about another $\lceil \log n \rceil$ Toffolis. For each we can use CNOTs to output a success flag on an ancilla qubit.

We can also output the value of $\binom{k-1}{2}$, $\binom{k}{2}$, or $\binom{k+1}{2}$ in another register. In this case we would need to apply an equality test between the result in our sum register and the result in this register, which again has a Toffoli complexity no larger than $2\log k$. There are also n Toffolis needed to sum the ones in x and no more than $3\lceil \log n \rceil$ Toffolis to check the number of ones.

Much of the complexity of the algorithm is due to the use of amplitude amplification to find the cliques. There has been much work on quantum algorithms for clique finding, but these algorithms are typically posed in terms of calls to an oracle for the graph, with a possibly large complexity for additional gates. What that means is that the complexity in terms of oracle calls is no more than $\mathcal{O}(n^2)$ to find all the edges, and then there can be a very large amount of postprocessing to find the cliques.

C. Amplifying the initial state

We aim to amplify the initial state so that we have the state with an equal superposition over cliques. The strategy is to initially estimate the amplitude, then apply the appropriate number of steps of amplitude amplification when we are preparing the state to estimate the size of the null eigenspace. It is possible to show that the complexity of estimating the amplitude is as given in the following Lemma.

Lemma 1 (Quantum amplitude estimation)**.** *Let U be a unitary and let $0 < a < 1$ be such that*

$$U|0,0\rangle = a|\psi_0,0\rangle + \sqrt{1-a^2}\,|\psi_1,1\rangle\,. \tag{17}$$

There exists a quantum algorithm which estimates a to within error ϵ with probability of error less than δ, using

$$N = \frac{\pi}{\epsilon}\sqrt{1+\alpha^2} = \frac{1}{2\epsilon}\ln(1/\delta) + \mathcal{O}(\epsilon^{-1}\ln\ln(1/\delta)) \tag{18}$$

calls to U or $U^\dagger$.

The proof for this Lemma is given in Appendix C. To see the value of ϵ needed, note that probability of success will be reduced to approximately $\sin^2((1\pm\epsilon/a)\pi/2)$ if we incorrectly choose the number of iterates in the amplitude amplification due to imprecision in estimating the amplitude. That translates to a probability of failure of the amplitude amplification of approximately $(\epsilon\pi/2a)^2$. For our application, the amplitude is approximately

$$\sqrt{1-\frac{1}{2c}}\sqrt{\frac{|\mathrm{Cl}_k(G)|}{\binom{n}{k}}}, \tag{19}$$

where the first factor comes from failure of the Dicke state preparation, and the second from the clique checking. For simplicity, in the following expressions for complexity we will omit the factor of $\sqrt{1-1/2c}$ which is close to 1. The amplitude estimation is needed because it typically will be unknown how many cliques there are $|\mathrm{Cl}_k(G)|$. Inaccuracy in the amplitude estimation translates to a probability for failure of the amplitude amplification due to using an incorrect number of steps.

In practice the "failure" of the amplitude amplification is not a major problem, because it can be combined into an uncertainty in estimation of the Betti number. That is, in the next step instead of estimating the Betti number relative to $|\mathrm{Cl}_k(G)|$, we will be estimating it relative to a value that may be increased by about a factor of $1/[1-(\epsilon\pi/2a)^2]$ (using the approximation of the sin function). If we want $(\epsilon\pi/2a)^2$ no more than a relative error r, then we should choose

$$\epsilon \leq \frac{2\sqrt{r}}{\pi}\sqrt{\frac{|\mathrm{Cl}_k(G)|}{\binom{n}{k}}}. \tag{20}$$

That means that the cost would be

$$\frac{\ln(1/\delta)}{\sqrt{r}}\frac{\pi}{4}\sqrt{\frac{\binom{n}{k}}{|\mathrm{Cl}_k(G)|}} \tag{21}$$

steps. In comparison, the number of steps of the amplitude amplification is approximately

$$\frac{\pi}{4}\sqrt{\frac{\binom{n}{k}}{|\mathrm{Cl}_k(G)|}}. \tag{22}$$

That is, the amplitude estimation is more costly by a factor of $\ln(1/\delta)/\sqrt{r}$.

This cost of the Dicke state preparation from (12) will be doubled in amplitude estimation and amplification when we account for the need to unprepare the Dicke state. As a result the Toffoli costs of each step of the amplitude estimation and amplification are as follows.

1. The preparation of the Dicke state takes

$$(\lceil\log(cn)\rceil+1)\,[n(\lceil\log(cn)\rceil+2)+2\lceil\log n\rceil] \tag{23}$$

   Toffolis.

2. The cost of reflection on the clique check is given by $6|E|+2\log k$.

D. Block-encoding the sparse Hamiltonian

Having constructed the sparse oracles in the previous section, we now implement a quantum circuit that block-encodes the sparse Hamiltonian. We use a similar principle as in [10], except here we are implementing the Dirac operator B_G rather than the combinatorial Laplacian. In that work it is shown that the Dirac operator for *all* Hamming weights and unrestricted by the cliques can be written as

$$B = \sum_{j=1}^{n}(a_j + a_j^\dagger), \tag{24}$$

where a_j and $a_j^\dagger$ are fermionic annihilation and creation operators on qubit j. Using the usual Jordan-Wigner representation that gives the Hamiltonian

$$\sum_{j=1}^{n} Z_1 \otimes Z_{j-1} \otimes X_j, \tag{25}$$

where the subscripts indicate the qubits that these operators act on (starting the numbering from 1). This is the core of the implementation of the complete Hamiltonian, and can easily be implemented by first preparing

an equal superposition state over n basis states, then applying the controlled string of Pauli operators as in Figure 9 of [28].

To understand the reason that the Pauli string encodes the matrix, note that ∂_k will remove a one from some location in the bit string x of Hamming weight $k+1$ and apply a sign according to the number of ones prior to that location. That can be achieved by applying an X in that location, and applying Z gates on all qubits prior to that location. We need a superposition of applying the X in all locations where there are ones. Moreover, we also want to apply $\partial_{k+1}^\dagger$ to a bit string of Hamming weight $k+1$. This involves flipping a zero to a one (which can be done with an X gate) and applying a sign according to the number of ones prior to that position. This can again be done using a string of Z gates. Now we want a superposition of performing X gates at all locations where there are ones, *and* X gates where there are zeros, which can be implemented by the above sum of Pauli strings.

Here we aim to block encode the matrix

$$B_G = \begin{bmatrix} 0 & \partial_{k-1}^G & 0 \\ \partial_{k-1}^{G\dagger} & 0 & \partial_k^G \\ 0 & \partial_k^{G\dagger} & 0 \end{bmatrix}. \tag{26}$$

The difference of this from the unrestricted case B in [10] is that it only acts on states with Hamming weight $k-1, k, k+1$, and gives zero otherwise. Similarly, it only gives states with Hamming weight in this range. Moreover, B_G is restricted to the clique subspace. That means it must give zero if the input state is not a clique, and must also not give any output states that are not cliques.

Next we provide a general method of constructing a qubiterate operator in cases where tests on the system state are required. The block encoding with the tests can be described as

$$(|0\rangle\langle 0| \otimes P)\, V\, (|0\rangle\langle 0| \otimes P) = |0\rangle\langle 0| \otimes B_G/\lambda, \tag{27}$$

where P is a projection on the system that tests the Hamming weight and cliques. We are adopting notation similar to Eq. (3) in [23], but replacing the identity with P to indicate that a projection is needed on the target system. We will assume V is Hermitian; if it is not we can construct a Hermitian V by block encoding it as $V \mapsto V \otimes |1\rangle\langle 0| + V^\dagger \otimes |0\rangle\langle 1|$ [29]. Similarly, we are writing B_G for the operator we aim to block encode, but this reasoning applies for a more general Hamiltonian H.

If $|k\rangle$ is an eigenstate of B_G with energy E_k and satisfying $P|k\rangle = |k\rangle$, then by definition we must have

$$V|0\rangle|k\rangle = \frac{E_k}{\lambda}|0\rangle|k\rangle + i\sqrt{1 - \left|\frac{E_k}{\lambda}\right|^2}\left|0k^\perp\right\rangle, \tag{28}$$

where $\left|0k^\perp\right\rangle$ is defined as a state such that

$$(|0\rangle\langle 0| \otimes P)\left|0k^\perp\right\rangle = 0. \tag{29}$$

Then we can define the qubiterate as

$$W := RV, \tag{30}$$

with

$$R := i\,(2\,|0\rangle\langle 0| \otimes P - I). \tag{31}$$

This is similar to that in [23], except we have included the projection P in the reflection operation. That is, we are applying the tests as part of the reflection, instead of applying them in the operation V.

Then we obtain

$$W|0\rangle|k\rangle = i\frac{E_k}{\lambda}|0\rangle|k\rangle + \sqrt{1 - \left|\frac{E_k}{\lambda}\right|^2}\left|0k^\perp\right\rangle. \tag{32}$$

It is also found that

$$W\left|\chi k^\perp\right\rangle = i\frac{E_k}{\lambda}\left|\chi k^\perp\right\rangle + \sqrt{1 - \left|\frac{E_k}{\lambda}\right|^2}|\chi\rangle|k\rangle. \tag{33}$$

Here we have corrected a minor error from [23] where there was an i appearing on the second term. See Appendix D for the derivation. Then it is easy to see that

$$\frac{1}{\sqrt{2}}\left(|0\rangle|k\rangle \pm \left|0k^{\perp}\right\rangle\right) \tag{34}$$

are eigenstates of W with eigenvalues $\pm e^{\pm i \arcsin(E_k/\lambda)}$. This is the usual relation for the eigenvalues of the qubitised operators, showing that this approach for constructing the walk operator works.

For our implementation here, V is just the controlled string of Pauli operators together with preparation of an equal superposition state. The reflection on the target system expressed by the projector can be implemented by computing an ancilla qubit flagging that the projection is satisfied (we have the appropriate Hamming weight range and cliques), reflecting on that qubit and the control qubits, then uncomputing the test. In some cases this can give a significant reduction in complexity over performing the test before and after V. If the ancilla qubits used to compute the tests are retained, then they can be erased with Clifford gates and measurements. For the application here that would be too costly in terms of ancilla qubits, so we incur the Toffoli cost of the test again in erasing the ancillas.

For the complexity of the implementation we have the following costs.

1. Preparing an equal superposition state over n basis states, which can be performed with complexity $4\lceil \log n \rceil + 1$ Toffolis [7], or just with Hadamards if n is a power of 2. This cost is incurred twice.
2. The controlled string of Pauli operators can be applied with Toffoli complexity $n-1$ using the method in Figure 9 of [28].
3. The Hamming weight can be computed with no more than n Toffolis and n ancilla qubits [26]. In that case we would not need to double the complexity for the reflection because the sum can be uncomputed with Cliffords. We could use $2n$ Toffolis with a logarithmic number of qubits [7], but in that case we would need to double the complexity for the uncomputation cost.
4. The complexity of outputting qubits with $\binom{k-1}{2}$, $\binom{k}{2}$, or $\binom{k+1}{2}$ is no more than $3\lceil \log n \rceil$. At the same time we can use the QROM to output a qubit which flags if the Hamming weight is outside the range. These qubits can be erased with Cliffords by retaining a logarithmic number of ancilla qubits.
5. As described above the cost of the reflection on the clique test is no more than $6|E| + 2\log k$.
6. Note that there is a reflection on the result of two tests, but this would correspond to a controlled-Z which is a Clifford gate.

The overall complexity is, assuming we use a logarithmic number of ancilla qubits,

$$6|E| + 5n + 11\log n + 2\log k + \mathcal{O}(1). \tag{35}$$

Because there is a linear combination of n Pauli strings, the value of λ for this block encoding is n. This value will be increased very slightly because of imperfect preparation of an equal superposition state in the method of [7]. That increase is normally less than one part in 1000, so will be ignored here.

E. Projection-based overlap estimation

In order to estimate the number of zero eigenvalues of the Hamiltonian, we project onto the zero eigenspace, then perform amplitude estimation. The projection can be approximated using a Chebyshev polynomial approach. First, recall that in the qubitisation the zero eigenvalue of the Hamiltonian is mapped to eigenvalues ± 1 of the qubitised operator. For the filter function on the phase ϕ of the eigenvalues of the walk operator, one can take

$$\tilde{w}(\phi) = \epsilon T_\ell\left(\beta \cos\left(\phi\right)\right) \tag{36}$$

for ϕ taking discrete values $\pi k/\ell$ for k from $-\ell$ to ℓ, and where $\beta = \cosh(\frac{1}{\ell}\cosh^{-1}(1/\epsilon))$. Taking the discrete Fourier transform of these values gives the window w_j such that

$$\tilde{w}(\phi) = \sum_{j=-\ell}^{\ell} w_j e^{ij\phi}. \tag{37}$$

Note that $\tilde{w}(\phi)$ is a function of $\cos\phi$, so $w_j = w_{-j}$. Moreover, we have values of j separated by 2. If ℓ is even, then we have even powers of $\cos\phi$, and therefore only even j. This means that it can be regarded as a polynomial in $2^{2i\phi}$. We can select between the qubitised walk step and its inverse by controlling on the reflection, so implementing a linear combination of unitaries may be performed with cost ℓ.

The peak for $\tilde{w}(\phi)$ will be at 0 and π, which is what is needed because the qubitised operator produces duplicate eigenvalues at phases of 0 and π. The width of the operator can be found by noting that the peak is for the argument of the Chebyshev polynomial equal to β, and the width is where the argument is 1, so $\beta\cos(\phi) = 1$. This gives us

$$\cosh(\tfrac{1}{\ell}\cosh^{-1}(1/\epsilon))\cos(\phi) = 1. \tag{38}$$

The gap in the Hamiltonian is $\lambda_{\min}$, which translates to a gap in the qubitised operator of $\arcsin(\lambda_{\min}/\lambda)$. Because the width of the peak should be equal to the gap, we can replace ϕ with $\arcsin(\lambda_{\min}/\lambda)$, and solving for ℓ gives

$$\ell = \frac{\cosh^{-1}(1/\epsilon)}{\cosh^{-1}(1/\sqrt{1-(\lambda_{\min}/\lambda)^2})} \leq \frac{\lambda}{\lambda_{\min}}\ln(2/\epsilon). \tag{39}$$

In the case that $\lambda = n$, the complexity in terms of calls to the block encoding is

$$\frac{n}{\lambda_{\min}}\ln(2/\epsilon). \tag{40}$$

To determine the appropriate value of ϵ to take, note that ϵ tells us the multiplying factor for amplitudes for states with eigenvalues outside the gap. The state starts with equal weighting on all eigenvalues, so ideally we should have the amplitude after filtering $\sqrt{\beta_{k-1}^G/|\mathrm{Cl}_k(G)|}$. If the state amplitudes outside the gap are multiplied by ϵ, then the error in the squared amplitude can be at most ϵ^2. This corresponds to an error in β_{k-1}^G of $\epsilon^2|\mathrm{Cl}_k(G)|$, or a relative error of $\epsilon^2|\mathrm{Cl}_k(G)|/\beta_{k-1}^G$.

F. Total complexity of algorithm

The Toffoli costs of the algorithm are as follows.

1. The preparation of the Dicke state has a leading order complexity

$$n\log^2 n + \mathcal{O}(n\log n) \tag{41}$$

   Toffolis.

2. The cost of checking cliques is given by $6|E| + 2\log k$ where we take into account the need to uncompute the result.

3. The cost of amplitude estimation is a number of iterations of steps 1 and 2 given as

$$\frac{\ln(1/\delta)}{\sqrt{r}}\frac{\pi}{4}\sqrt{\frac{\binom{n}{k}}{|\mathrm{Cl}_k(G)|}}. \tag{42}$$

4. The cost of amplitude amplification of the cliques is given by approximately $(\pi/4)\sqrt{\binom{n}{k}/|\mathrm{Cl}_k(G)|}$ of iterations of steps 1 and 2.

5. The walk step for the qubitisation needs $6|E| + 5n + 11\log n + 2\log k + \mathcal{O}(1)$ Toffolis.

6. For the filtering there are $\frac{n}{\lambda_{\min}}\ln(2/\epsilon)$ calls to the block encoding with costs in item 5 above.

7. Lastly, we need to perform amplitude estimation on the entire procedure, using $\approx \log(1/\delta)/2\epsilon$ calls to the amplitude amplification in 4 and filtering in 6.

To give the leading-order complexities, the combined cost of steps 1 and 2 is

$$6|E| + n\log^2 n + \mathcal{O}(n\log n). \tag{43}$$

To distinguish the ϵ, δ and r (relative error) needed in different steps we will use subscripts. The cost of amplitude estimation is then approximately

$$\frac{\ln(1/\delta_1)}{\sqrt{r_1}}\frac{\pi}{4}\sqrt{\frac{\binom{n}{k}}{|\mathrm{Cl}_k(G)|}}(6|E| + n\log^2 n). \tag{44}$$

This is expected to be a trivial cost in the overall algorithm, because the amplitude amplification is performed many more times.

For the remainder of the algorithm, we have an initial cost of

$$\frac{\pi}{4}\sqrt{\frac{\binom{n}{k}}{|\mathrm{Cl}_k(G)|}}(6|E| + n\log^2 n) \tag{45}$$

for the amplitude amplification for the initial state. Then there is a cost for the block encoding of

$$6|E| + 5n + \mathcal{O}(\log n), \tag{46}$$

for each step. Multiplying by the number of steps needed for filtering, there is a cost

$$\frac{n}{\lambda_{\min}}\ln(2/\epsilon_3)[6|E| + 5n + \mathcal{O}(\log n)]. \tag{47}$$

To determine the appropriate value of ϵ_3 to take, note that we are measuring a kernel of size β^G_{k-1} as compared to an overall dimension of $|\mathrm{Cl}_k(G)| \gg \beta^G_{k-1}$. The relative accuracy in the estimation of β^G_{k-1} will therefore be about $\epsilon_3^2|\mathrm{Cl}_k(G)|/\beta^G_{k-1}$ as explained above at the end of Section III E. If we aim for relative accuracy r_3, then we have complexity

$$\frac{n}{2\lambda_{\min}}\ln\left(\frac{4|\mathrm{Cl}_k(G)|}{r_3\beta^G_{k-1}}\right)(6|E| + 5n + \mathcal{O}(\log n)). \tag{48}$$

Lastly, the amplitude estimation on the entire procedure needs a number of repetitions

$$\frac{\ln(1/\delta_2)}{2\epsilon_2}. \tag{49}$$

But, this amplitude estimation is on a number of steps corresponding to a reflection requiring both the forward and reverse calculations. That introduces a further factor of 2, so we should use

$$\frac{\ln(1/\delta_2)}{\epsilon_2}. \tag{50}$$

Next, ϵ_2 corresponds to an accuracy of estimating a ratio $\sqrt{\beta^G_{k-1}/|\mathrm{Cl}_k(G)|}$. If we want a relative accuracy r_2, then the error propagation formula gives

$$r_2 = \frac{\Delta\beta^G_{k-1}}{\beta^G_{k-1}} = \frac{\epsilon_2}{\beta^G_{k-1}}\left(\frac{d}{d\beta^G_{k-1}}\sqrt{\frac{\beta^G_{k-1}}{|\mathrm{Cl}_k(G)|}}\right)^{-1} = 2\epsilon_2\sqrt{\frac{|\mathrm{Cl}_k(G)|}{\beta^G_{k-1}}}, \tag{51}$$

where $\Delta\beta^G_{k-1}$ is uncertainty in β^G_{k-1}. In terms of r_2, the number of repetitions becomes

$$2\frac{\ln(1/\delta_2)}{r_2}\sqrt{\frac{|\mathrm{Cl}_k(G)|}{\beta^G_{k-1}}}. \tag{52}$$

Applying this to the complexity required for each step, we get a complexity of approximately

$$\frac{\ln(1/\delta_2)}{r_2}\sqrt{\frac{|\mathrm{Cl}_k(G)|}{\beta_{k-1}^G}}\left[\frac{\pi}{2}\sqrt{\frac{\binom{n}{k}}{|\mathrm{Cl}_k(G)|}}(6|E|+n\log^2 n)+\frac{n}{\lambda_{\min}}\ln\left(\frac{4|\mathrm{Cl}_k(G)|}{r_3\beta_{k-1}^G}\right)(6|E|+5n)\right]. \tag{53}$$

Comparing this to the amplitude estimation cost in (44) the primary difference is the factor of

$$\sqrt{\frac{|\mathrm{Cl}_k(G)|}{\beta_{k-1}^G}} \tag{54}$$

here. There is another difference in that the amplitude estimation cost has the factor $1/\sqrt{r_1}$ rather than $1/r_2$, so the scaling in terms of relative error is improved. In cases where the number of cliques $|\mathrm{Cl}_k(G)|$ is much larger than the Betti number β_{k-1}^G, then the amplitude estimation cost is trivial.

We will have a total probability of failure $\delta=\delta_1+\delta_2$ due to the two amplitude estimations, and a total relative error $r_1+r_2+r_3$. In order to reduce the complexity, we can use the fact that the cost of the initial amplitude estimation is much smaller, so we can take δ_1 and r_1 smaller without much impact on the overall complexity. The r_3 appears inside a logarithm, so can be taken to be smaller than r_3.

To give the scaling of the complexity in a simpler way, we can simply ignore the amplitude estimation complexity, and replace δ_2 with δ, and replace both r_2 and r_3 with r (since r_3 can be taken as for example $r/20$ without much impact on the overall complexity). We will also omit terms of complexity kn or n as compared to $|E|$. That then gives

$$T(G,k,r,\delta):=6|E|\frac{\ln(1/\delta)}{r}\sqrt{\frac{|\mathrm{Cl}_k(G)|}{\beta_{k-1}^G}}\left[\frac{\pi}{2}\sqrt{\frac{\binom{n}{k}}{|\mathrm{Cl}_k(G)|}}+\frac{n}{\lambda_{\min}}\ln\left(\frac{4|\mathrm{Cl}_k(G)|}{r\beta_{k-1}^G}\right)\right], \tag{55}$$

where $T(G,k,r,\delta)$ gives the required number of Toffoli gates to estimate, with precision parameters r,δ, the $(k-1)$-th order Betti number of a graph G with n nodes, $|E|$ edges and a Laplacian with gap $\lambda_{\min}$.

This expression for the complexity is in terms of the relative accuracy r. Alternatively, if we aimed for a given absolute accuracy α, then $\alpha=r\beta_{k-1}^G$, and so the expression for the complexity becomes

$$T(G,k,r,\delta):=6|E|\frac{\ln(1/\delta)}{\alpha}\sqrt{|\mathrm{Cl}_k(G)|\beta_{k-1}^G}\left[\frac{\pi}{2}\sqrt{\frac{\binom{n}{k}}{|\mathrm{Cl}_k(G)|}}+\frac{n}{\lambda_{\min}}\ln\left(\frac{4|\mathrm{Cl}_k(G)|}{\alpha}\right)\right]. \tag{56}$$

We now discuss the complexity of just the first term in the square brackets, which corresponds to the state preparation cost rather than the filtering cost. That cost will be dominant if the gap is large, though it must be emphasised that the gap will be small in many cases. This first term for the cost gives

$$T(G,k,r,\delta)=3\pi|E|\frac{\ln(1/\delta)}{r}\sqrt{\frac{\binom{n}{k}}{\beta_{k-1}^G}}. \tag{57}$$

If we are aiming for a given absolute accuracy α in β_{k-1}^G, then the complexity would be

$$T(G,k,r,\delta)=3\pi|E|\frac{\ln(1/\delta)}{\alpha}\sqrt{\binom{n}{k}\beta_{k-1}^G}. \tag{58}$$

The complexity is now larger for large Betti number β_{k-1}^G. The reason for this is that the amplitude estimation is estimating the *square root* of β_{k-1}^G. The square root has a small derivative for large values of β_{k-1}^G, making it more difficult to estimate the Betti number with small absolute error. Again, note that the last three expressions above are only for the state preparation cost, without the filtering cost.

To compare to the complexity of classical approaches, an exact diagonalisation approach would tend to scale as $\binom{n}{k}^2$, whereas approximate schemes scale as $\binom{n}{k}$. Thus the quantum algorithm would give approximately a square-root speedup over these classical algorithms if β_{k-1}^G is on the order of a constant. On the other hand, for graphs with large β_{k-1}^G, a speedup that is greater than a square root can be obtained.

IV. REGIMES FOR QUANTUM SPEED-UP

In this section, we ask if there exist regimes where our quantum algorithm offers a significant speedup over the best classical algorithms. The aim is to compute to relative error the $(k-1)^{\text{th}}$ Betti number of the clique complex of a graph G. Say G has n nodes, $|E|$ edges, r is the desired multiplicative error, and $\lambda_{\min}$ is the spectral gap of the combinatorial Laplacian $\Delta_{k-1}^G = \partial_{k-1}^{G\dagger}\partial_{k-1}^G + \partial_k^G \partial_k^{G\dagger}$. To simplify the arguments, we will represent the quantum complexity of this problem as

$$T_q = \widetilde{\mathcal{O}}\left(\frac{n\,|E|}{r\,\lambda_{\min}}\sqrt{\frac{1}{\beta_{k-1}}\binom{n}{k}}\right). \tag{59}$$

Comparing to Eq. (55), this will asymptotically upper bound both terms up to log factors.

For a rough estimate of the cost of computing the Betti number classically, one could use $|\mathrm{Cl}_k(G)|$ (i.e., the number of k-cliques) or $\binom{n}{k}$. The reason is that classical algorithms typically start by constructing a list of k-cliques, and afterwards compute the nullity of the combinatorial Laplacian or boundary operator. The cost of this second step (i.e., estimating the nullity of the combinatorial Laplacian or boundary operator), scales at best linearly in size of the matrix $|\mathrm{Cl}_k(G)|$ [30]. On the other hand, the first step (i.e., listing all k-cliques) can be done using a brute force search at cost $\binom{n}{k}$. There are more efficient algorithms for listing cliques, though the complexity tends to be dependent on the properties of the graph. However, $|\mathrm{Cl}_k(G)|$ always lower bounds the cost of listing all the k-cliques. Therefore, $|\mathrm{Cl}_k(G)|$ and $\binom{n}{k}$ can be considered to be lower and upper bounds on the scaling of the classical complexity, respectively. In conclusion, the best classical algorithms for this problem have scaling lower bounded by

$$T_c = \Omega\left(|\mathrm{Cl}_k(G)|\right). \tag{60}$$

Recall $\mathrm{Cl}_k(G)$ is the set of cliques of size k, which form the $(k-1)$-simplices of the simplicial complex. Classical algorithms have extra factors in the complexity, such as $1/r^2$ dependence on the required precision, that introduce orders of magnitude over this lower bound.

The quantum algorithm will offer a speedup on instances where β_{k-1} is large, and where $\lambda_{\min}$ is not too small. We can remove the dependence on $\lambda_{\min}$ if we instead focus on computing an approximate Betti number, in the following sense.

Definition 1. *The δ-approximate k^{th} Betti number is $B_k^\delta = \dim\{v \in \mathcal{H}_{k-1}^G : \frac{v^\dagger \Delta_k v}{v^\dagger v} \le \delta\}$. Note $B_k^0 = \beta_k$, and in general $B_k^\delta \ge \beta_k$.*

The same quantum algorithm computes B_{k-1}^δ to relative error with cost

$$T_q = \widetilde{\mathcal{O}}\left(\frac{n\,|E|}{r\,\delta}\sqrt{\frac{1}{B_{k-1}^\delta}\binom{n}{k}}\right). \tag{61}$$

A. A family of graphs with large Betti numbers and large spectral gaps

Let $K(m,k)$ be the k-partite complete graph, where each partition contains m vertices. That is, $K(m,k)$ consists of k clusters, each with m vertices; there are no edges within clusters, but all edges between clusters are included. Note $K(m,1)$ is a collection of m points with no edges.

$K(m,k)$ gives a useful example of a clique complex with a high Betti number [31]. It also has a Laplacian with a large spectral gap.

Proposition 1. *The $(k-1)^{th}$ Betti number of (the clique complex of) $K(m,k)$ is*

$$\beta_{k-1} = (m-1)^k. \tag{62}$$

Proposition 2. *The combinatorial Laplacian $\Delta_{k-1} = \partial_{k-1}^\dagger \partial_{k-1} + \partial_k \partial_k^\dagger$ of (the clique complex of) $K(m,k)$ has spectral gap*

$$\lambda_{\min} = m. \tag{63}$$

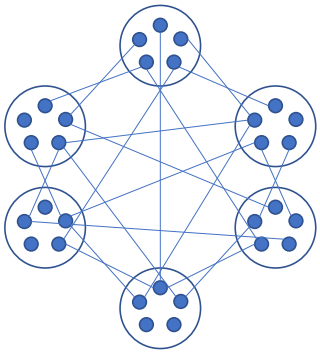

FIG. 1. The graph $K(5,6)$.

We prove these in Appendix E using techniques from simplicial homology. A further useful fact is that

$$|\mathrm{Cl}_k(K(m,k))| = m^k. \tag{64}$$

Having set up all of the above, let us now go over various values for k to analyze the quantum speedup attained on this family of graphs. We will compute the runtime of the quantum algorithm and compare it with the correponding number of cliques, which is the runtime of the classical algorithm.

First, set $k = n/m$ for some integer $m \geq 2$ that divides n and consider $K(m, n/m)$. Proposition 1 gives $\beta_{k-1} = (m-1)^{n/m}$, and $\lambda_{\min} = m$ by Proposition 2. Ignoring factors polynomial in n, Stirling's approximation gives $\binom{n}{k} \sim \left(\frac{m^{1+1/m}}{m-1}\right)^n$. Combining this we find, ignoring factors polynomial in n

$$T_q \sim \left(\frac{m^{1+1/m}}{(m-1)^{1+1/m}}\right)^{n/2}$$

whereas by Eq. (64),

$$T_c \sim m^{n/m}.$$

In conclusion, for this value of k the quantum algorithm achieves a polynomial speedup whose degree depends on the choice of m. The largest ratio between the exponents occurs at $m = 6$, where $\left(\frac{m}{m-1}\right)^{1/2+1/2m} = 1.11$ and $m^{1/m} = 1.35$. At the moment it is unclear whether there exist graphs that have a large enough Betti number such that the quantum speedup is exponential (superpolynomial speedups are discussed below).

Secondly, set $k = n^\alpha$ for some $\alpha \in (0,1)$ and consider $K(n^{1-\alpha}, n^\alpha)$. Proposition 1 gives $\beta_{k-1} \sim n^{(1-\alpha)n^\alpha}$, and Proposition 2 gives $\lambda_{\min} = n^{1-\alpha}$. Ignoring factors polynomial in n, Stirling's approximation gives $\binom{n}{k} \sim \exp\left(n^\alpha(\log n^{1-\alpha} + 1)\right)$. Combining this we find, ignoring factors polynomial in n

$$\log T_q \sim n^\alpha$$

whereas by Eq. (64),

$$\log T_c \sim n^\alpha \log n^{1-\alpha}.$$

In conclusion, for this value of k the quantum algorithm achieves a superpolynomial speedup.

Next, set $k = c \log n$ for some constant c and consider $K(n/c\log n, c\log n)$. Proposition 1 gives $\beta_{k-1} \sim (n/c\log n)^{c\log n}$, and Proposition 2 gives $\lambda_{\min} = n/c\log n$. We have $\binom{n}{k} \leq n^{c\log n}/(c\log n)!$. Combining this and using Stirling's approximation, we find

$$T_q \sim \frac{(c\log n)^{c\log n}}{(c\log n)!} = \mathcal{O}(n^c)$$

whereas by Eq. (64),

$$T_c = \mathcal{O}\left(\left(\frac{n}{c\log n}\right)^{c\log n}\right).$$

In conclusion, for this value of k the quantum algorithm achieves a superpolynomial speedup.

Finally, set k constant and consider $K(n/k, k)$. Proposition 1 gives $\beta_{k-1} \sim n^k$, and Proposition 2 gives $\lambda_{\min} = n/k = \Omega(n)$. Moreover, $\binom{n}{k} \sim n^k$. Combining this we find

$$T_q = \mathcal{O}(n^2),$$

whereas by Eq. (64),

$$T_c = \mathcal{O}(n^k).$$

The $\mathcal{O}(n^2)$ left over for the quantum cost arises from the factor of $|E|$.

Next we provide numerical results for the Toffoli complexity as a function of n and k. For these calculations we have made a number of adjustments to our simplified expressions in order to provide more accurate results. In particular we compute the integral of the Kaiser window rather than using the asymptotic expression, as well as including the Dicke state preparation cost and the initial amplitude estimation cost for the number of steps needed for the state preparation.

The results are as given in Fig. 2 as a function of n for a range of values of k. It can be seen that the cost of the quantum algorithm for a given k scales approximately as n^2, with the cost scaling primarily coming from the number of edges in the graph. The classical cost given as the number of k-cliques or $\binom{n}{k}$ has similar scaling, which is considerably worse than for the quantum algorithm, and is much worse for larger values of k, as expected from the analysis above.

For the example of $n = 256, k = 16$ the quantum cost is approximately 80 billion Toffolis, which is comparable to gate counts for classically intractable instances of quantum chemistry [32]. In contrast, the number of cliques is about 2×10^{19}, and $\binom{n}{k} \approx 10^{25}$. These numbers are sufficiently large that it should be classically intractable. For example, just storing the vector would be beyond the storage capacity of supercomputers. With $n = 180, k = 12$ the quantum cost is reduced to 10 billion, but the classical cost is reduced by about 5 orders of magnitude, as indicated by the number of cliques as about 1.3×10^{14}, or $\binom{n}{k} \approx 1.7 \times 10^{18}$.

B. Erdős-Rényi graphs

The family of graphs in Section IV A is specifically constructed to have high Betti number and large spectral gap. One might wonder what speedups are *generically* possible. To shed light on this question, we examine the Erdős-Rényi family of random graphs.

The Erdős-Rényi random graph $G(n, p)$ has n vertices, and each of the $\binom{n}{2}$ edges is present i.i.d. with probability p. In [33], the following theorem is established.

Theorem 1. *Let $p = n^\alpha$. If $-1/k < \alpha < -1/(k+1)$, then*

$$\frac{\beta_k}{\binom{n}{k+1} p^{\binom{k+1}{2}}} \to 1 \quad \textit{almost surely}$$

On the other hand, if $\alpha < -1/k$ or $\alpha > -1/(2k+1)$, then $\beta_k \to 0$ almost surely.

Taking $p = n^{-1/(k+\frac{1}{2})}$ gives $\beta_k \sim \binom{n}{k+1} n^{-k/2}$ almost surely. Ignoring the factor $n|E|/\lambda_{\min}$, our quantum algorithm can compute the k^{th} Betti number in time scaling as $T_q \sim n^{k/4}$ for constant k.

C. Rips complexes

One of the main applications of topological data analysis is to Rips complexes induced by finite-dimensional data in $\mathbb{R}^d$. This is another shortcoming of the graph family from Section IV A - they are defined as abstract

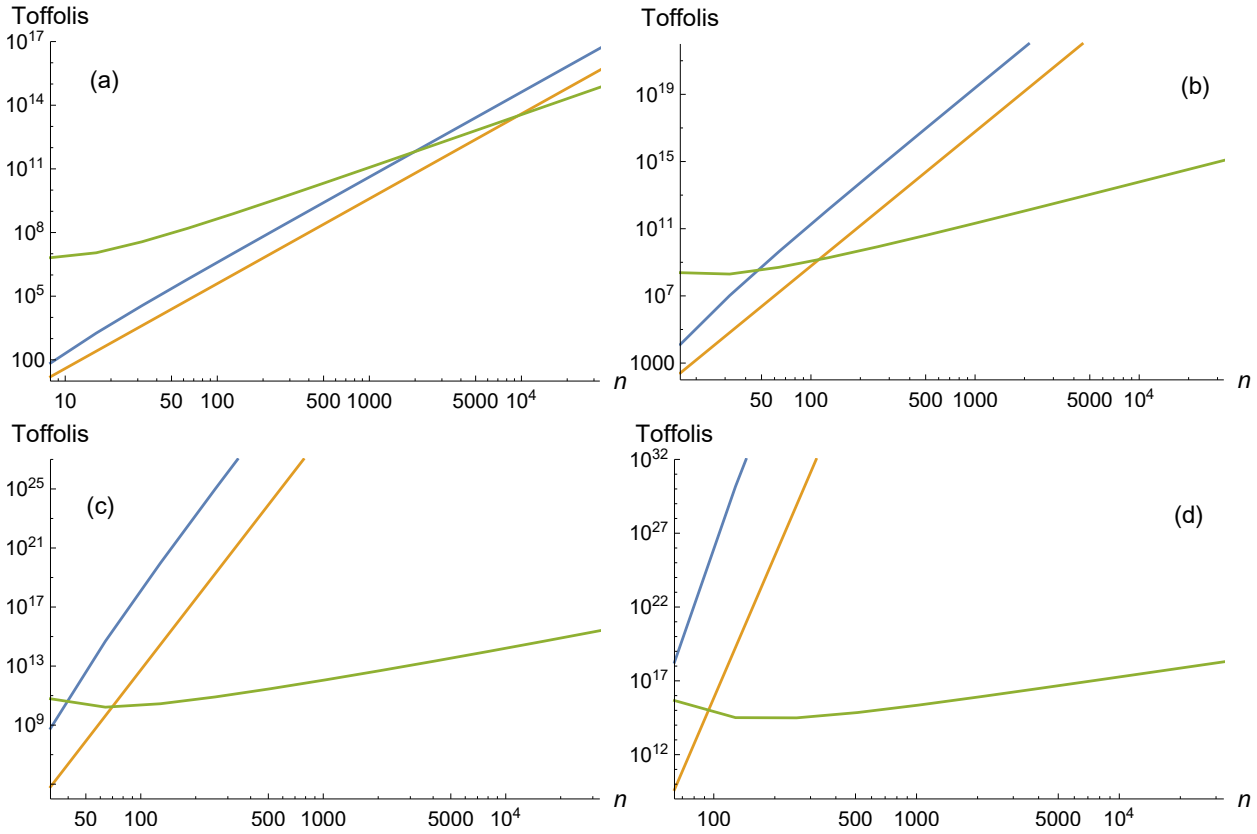

FIG. 2. The Toffoli counts for the quantum algorithm for the Betti number of $K(n, k)$ as a function of n for $k = 4$ (a), $k = 8$ (b), $k = 16$ (c), and $k = 32$ (d). The lines are green for the quantum complexity, blue for $\binom{n}{k}$, and orange for the number of cliques m^k. The values of δ and r are held constant at $1/20$. For the relative precision required r, the value of r_2 (filtering error) is taken to be $\neq 20$, and r_3 (the amplitude estimation error) is taken to be $r \times 0.95$.

graphs, rather than being induced from finite-dimensional data. But are such speedups possible for Rips complexes? Unfortunately, there are results which prevent these large speedups.

It is shown in [34] that, for any fixed k and d

$$\max_{S \subset \mathbb{R}^d : |S| = n} \frac{\beta_k(\mathcal{R}_\epsilon(S))}{n^k} \to 0 \text{ as } n \to \infty$$

In [35], the author studies a setting where n data points are drawn from a fixed underlying probability measure on $\mathbb{R}^d$. This is arguably the setting of interest in topological data analysis. They show that the Betti numbers of the derived Rips complexes have three 'phases' depending on the scale ϵ. (Recall that we include an edge if two points are within distance ϵ.) For small $\epsilon = o(n^{-1/d})$, called the subcritical phase, the Betti numbers vanish asymptotically. Intuitively the complex is highly disconnected, since we are below the percolation threshold. There is a critical phase $\epsilon \sim n^{-1/d}$, where the Betti numbers will scale linearly $\beta_k \sim n$. Then for large $\epsilon = \omega(n^{-1/d})$, in the supercritical phase, the Betti number grows sublinearly $\beta_k = o(n)$. Thus in all regimes, the Betti number grows at most linearly in the number of points. This is of course far from the n^k scaling needed for superpolynomial speedup.

However, it is possible to construct a Rips complex with large Betti number and large spectral gap, even in $\mathbb{R}^2$ [34]. We describe such a Rips complex here.

Construct $S \subset \mathbb{R}^2$ as follows. Let $m = n/2k$, $\theta = \pi/k$, and $\delta = n^{-4}$. For $i = 1, \ldots, k$, let $x_i^+ = (1/2, i\delta)$ and $x_i^- = (-1/2, i\delta)$. Let $S_0 = \{x_1^+, \ldots, x_m^+, x_1^-, \ldots, x_m^-\}$. For $j = 1, \ldots, k-1$, construct S_j by rotating S_0 about the origin by an angle $j\theta$. Then finally $S = S_0 \cup \cdots \cup S_{k-1}$. We will take the Rips complex $\mathcal{R}_1(S)$ with $\epsilon = 1$. $\{x_1^+, \ldots, x_m^+\}$ and $\{x_1^-, \ldots, x_m^-\}$ become m-simplices. There is an edge (x_i^+, x_i^-) for every i, but no edges (x_i^+, x_j^-) for $i \neq j$. Due to the small value of δ, each S_i is completely connected to every other S_j.

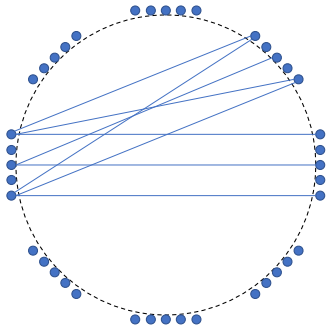

FIG. 3. The Rips complex $\mathcal{R}_1(S)$.

Proposition 3. *The $(2k-1)^{th}$ Betti number of $\mathcal{R}_1(S)$ is*

$$\beta_{2k-1}(S) = (m-1)^k = \left(\frac{n}{2k} - 1\right)^k.$$

Proposition 4. *The combinatorial Laplacian $\Delta_k = \partial_k^\dagger \partial_k + \partial_{k+1}\partial_{k+1}^\dagger$ of $\mathcal{R}_1(S)$ has constant spectral gap $\lambda_{\min}$.*

We prove these in Appendix E using techniques from simplicial homology. Our quantum algorithm can compute the k^{th} Betti number in time scaling as $T_q \sim n^{3+k/4}$ for constant k.

D. Randomized classical algorithms for Betti number estimation

While the previous discussion shows there are cases where quantum algorithms can provide exponential advantages with respect to deterministic classical algorithms for TDA, the question of how randomized classical algorithms perform in this setting is comparably understudied. Here we show that, perhaps surprisingly, there exists a randomized classical algorithm that under appropriate assumptions, can compute normalized Betti numbers in the clique dense case using a polynomial number of operations. This shows that the sufficient conditions needed for the quantum algorithm to provide an advantage are more subtle than anticipated and that simply having a high-dimensional vector space does not necessarily guarantee a super-polynomial speedup.

The main idea of our algorithm is to use imaginary-time evolution to create a projector onto the kernel of $\Delta_{k-1}^G = \partial_{k-1}^{G\dagger}\partial_{k-1}^G + \partial_k^G \partial_k^{G\dagger}$. More specifically, we focus on the Dirac operator B_G and look at simulating its imaginary time dynamics of its square using path integral Monte-Carlo. We further simplify this approach by taking $\tilde{B}_G$ to be an analogous operator to B_G except now we use an energy penalty to penalize any configuration that is not a clique or of the correct parity. In particular,

$$\widetilde{B_G^2} = B_G^2 + \gamma_{\min}(1-P), \tag{65}$$

where P as before is the projector onto the states of appropriate Hamming weight and configurations that correspond to a clique and $\gamma_{\min}$ is an upper bound on the spectral gap of B_G^2 which coincides with the second smallest eigenvalue of the combinatorial Laplacian. Further, it is easy to see that if a vector is in the kernel of $\widetilde{B_G^2}$ it is also in the kernel of B_G. Following the same reasoning as before, as B_G is Hermitian so is $\tilde{B}_G$ and thus it has a complete set of orthonormal eigenvectors. This implies that any unit vector $|\psi\rangle$ which is supported on $\mathcal{H}_k^G$ can be decomposed as

$$|\psi\rangle = \cos(\theta)\,|\psi_g\rangle + \sin(\theta)\,|\psi_b\rangle\,, \tag{66}$$

where $|\psi_g\rangle$ is the projection of $|\psi\rangle$ onto the kernel of $\widetilde{B_G^2}$ and $|\psi_b\rangle$ is its orthogonal complement. Then

$$e^{-\widetilde{B_G^2}t}\,|\psi\rangle = \cos(\theta)\,|\psi_g\rangle + e^{-\widetilde{B_G^2}t}\sin(\theta)\,|\psi_b\rangle\,. \tag{67}$$

Let $\widetilde{B_G^2}\,|\lambda_\mu\rangle = \lambda_\mu\,|\lambda_\mu\rangle$ such that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{d_k}$, where $d_k = \binom{n}{k}$. The operator $\widetilde{B_G^2}$ is self adjoint and it is positive semi-definite and thus

$$\langle\psi|\, e^{-B_G t}\,|\psi\rangle = \cos^2(\theta) + \sin^2(\theta)\,\langle\psi_b|\, e^{-\Delta_k^G t}\,|\psi_b\rangle \leq \cos^2(\theta) + \sin^2(\theta) e^{-\gamma t}. \tag{68}$$

If we then pick

$$t \geq \log(1/\epsilon)/\gamma_{\min}, \tag{69}$$

the expectation value will be at most $\cos^2(\theta) - O(\epsilon)$.

If $|\psi\rangle$ is chosen such that it is a column of a Haar random unitary over the constrained parity subspace $\mathcal{H}_k^G$, the expectation value will be

$$\mathbb{E}_{\text{Haar}}(\cos^2(\theta)) = \frac{\beta_{k-1}}{d_k}. \tag{70}$$

Thus performing imaginary time evolution and a Haar expectation value will give the required normalized Betti number.

The remaining question centers around whether the imaginary time evolution can be performed on a classical computer in polynomial time. First, let us consider a decomposition of the Hamiltonian of the form

$$\widetilde{B_G^2} = \sum_{p=1}^{D} c_p H_p \tag{71}$$

where each H_p is one-sparse and unitary and hence the eigenvalues of each are λ_{p_i,ν_i} in $\pm c_p$ where ν_i is an index of the eigenvalue and p_i is the index of the Hamiltonian. The Jordan-Wigner decomposition on B provides such a decomposition and the projector P can always be written as a sum of a reflection over computational basis states and an identity gate, which provides an efficient decomposition into one-sparse Hermitian and unitary terms.

With this decomposition in hand, we focus on using a path-integral Monte-Carlo simulation of $\exp(-\widetilde{B_G^2}t)$. The path integral expansion works by first breaking up $e^{-\widetilde{B_G^2}t}$ into r timeslices, Trotterizing over the matrices in a one-sparse decomposition of $\widetilde{B_G^2}$, and then expanding each one-sparse matrix in its eigenbasis. Since one sparse matrices can be efficiently diagonalized, this process is classically efficient. As notation, let Γ denote a particular path of eigenvectors in the path integral representation, $W(\Gamma)$ be the product of overlaps between the eigenvalues and λ_{p_i,Γ_i} be the eigenvalue corresponding to the eigenvector that appears in the i^{th} step in the path Γ. Finally, let $\Pr(\Gamma)$ be a probability distribution from which the paths are drawn that can be chosen to reduce the variance (as is standard in importance sampling). We show in Appendix F that taking the Haar-expectation of the result leads to

$$\mathbb{E}_{\text{Haar}}(\cos^2(\theta)) = \frac{1}{d_k}\mathbb{E}_\Gamma\left(\frac{\exp\left(-\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1}\lambda_{p_i,\Gamma_i} t/2r\right) W(\Gamma)\delta_{k_1,k_{2rD}}}{\Pr(\Gamma)}\right). \tag{72}$$

We then average over a finite ensemble of these random paths to estimate the expectation value drawn from an appropriate probability distribution. We propose, for general purposes, a Metropolis-Hastings based algorithm for selecting appropriate paths in the decomposition that are unlikely to have zero values of W. More specifically, the algorithm works by drawing an initial eigenstate of the first term in the one-sparse decomposition of the Dirac operator B_G uniformly. Then a path is drawn by transitioning to one of the two possible connected eigenstates for it randomly. As the terms are Hermitian by assumption, all eigenvalues at each step in the path integral are the same up to a sign. This means that there are only two choices when constructing a path: either we choose to traverse the positive eigenvalue or the negative eigenvalue. Thus each path can be described using $O(\log(d_k)rD)$ bits. A path that has non-zero overlaps between the neighboring eigenstates can then be selected in $O(\log(d_k)rD)$ time. This is used as an initial guess that is improved using Metropolis-Hastings, wherein the probability of transitioning between two randomly chosen paths $\Gamma^{(a)}$ and $\Gamma^{(b)}$ is:

$$P(\Gamma^{(b)}|\Gamma^{(a)}) = \frac{\exp\left(-2\lambda_{p_1,\Gamma_1^{(b)}}t/r - \sum_{i=2}^{2rD-1}\lambda_{p_i,\Gamma_i^{(b)}}t/r\right)}{\exp\left(-2\lambda_{p_1,\Gamma_1^{(a)}}t/r - \sum_{i=2}^{2rD-1}\lambda_{p_i,\Gamma_i^{(a)}}t/r\right)}. \tag{73}$$

The equilibrium distribution leads to a thermal distribution over the path K with

$$\Pr(\Gamma) = \frac{\exp\left(-2\lambda_{p_1,\Gamma_1}t/r - \sum_{i=2}^{2rD-1}\lambda_{p_i,\Gamma_i}t/r\right)\delta_{\Gamma\in S_\Gamma}}{\sum_{\Gamma\in S_\Gamma}\exp\left(-2\lambda_{p_1,\Gamma_1}t/r - \sum_{i=2}^{2rD-1}\lambda_{p_i,\Gamma_i}t/r\right)}, \tag{74}$$

where S_Γ is the set of all paths. The number of such updates needed to achieve this distribution (within fixed error) scales as $O(1/\gamma_M)$ where γ_M is the gap of the Markov chain. We show in Appendix F that, provided the gap is large, this distribution can be efficiently sampled from and forms a good choice for the importance distribution for the paths that minimizes the variance over the paths.

We ultimately find that the number of arithmetic operations needed to estimate the ratio of the kernel to the size of the set of all k-simplices within additive error ϵ is, assuming that the sample variance in the estimates yielded by Algorithm 1 is σ^2, is in

$$\widetilde{O}\left(\frac{\sigma^2}{\epsilon^2}\left(\frac{|E|\binom{n}{k}}{|\mathrm{Cl}_k(G)|} + \frac{D^4}{\gamma_M}\frac{\kappa^3}{\epsilon}\left(\log(d_k)D^{-2} + \frac{\kappa^3}{\epsilon}\right)\right)\right) \tag{75}$$

where κ is the ratio of the largest eigenvalue to the smallest non-zero eigenvalue, which plays the role of the condition number of the combinatorial Laplacian, and $|E|$ is the size of the edge set. This shows that even in cases where the dimension is exponentially large, we can use path integration to estimate the ratio of the dimension of the kernel of the combinatorial Laplacian to the dimension d_k of the space of $k-1$-simplices using a number of operations that scales polynomially with the dimension provided that the standard deviation D, σ, κ and γ_M^{-1} and the inverse clique density are at most polynomially large in n. We show, under much more restrictive assumptions, that σ can also be polynomially large in Appendix F.

The key point behind this dequantization result is that while an exponentially large dimension is a necessary condition for an exponential speedup for quantum TDA, it is not a sufficient condition. This implies that further work is needed in order to understand when, and even if, quantum algorithms can provide truly exponential advantages relative to all classical randomized algorithms for TDA.

Algorithm 1: Classical randomized algorithm for Betti number computation.

Data: $k \rhd\ 0, n \rhd\ 0, N_{\text{samp}} > 0$, $t \geq 0$, $r \geq 0$, a function $\Pr(\Gamma)$ which assigns a non-zero probability to each vector $\Gamma \in \mathbb{R}^{2rD}$, a function $W(\Gamma) = \langle\lambda_{p_1,\Gamma_1}|\lambda_{p_2,\Gamma_2}\rangle\cdots\langle\lambda_{p_{2rD-1},\Gamma_{2rD-1}}|\lambda_{p_1,\Gamma_1}\rangle$ where $|\lambda_{p_j,\Gamma_j}\rangle$ is the Γ_j^{th} eigenvector of the one sparse matrix U_{p_j}.

Result: Estimate $\bar{E}$ which is an unbiased estimator of β_{k-1}/d_k

for *q from* 1 *to* N_{samp} **do**

  $\Sigma \leftarrow$ a set of k points encoded as an integer

  while Σ is not a $(k-1)$-simplex **do**

    $\Sigma \leftarrow$ a random set of k points encoded as an integer

  end

  Draw a vector $\Gamma = [\Sigma, \Gamma_2, \ldots, \Gamma_{2rD}]$ from the probability distribution $\Pr(\Gamma)$.

  $E_q \leftarrow \frac{1}{d_k}\left(\frac{\exp\left(-\lambda_{p_1,\Gamma_1}t/r - \sum_{i=2}^{2rD-1}\lambda_{p_i,\Gamma_i}t/2r\right)W}{\Pr(\Gamma)}\right)$

end

$\bar{E} \leftarrow \frac{1}{N_{\text{samp}}}\sum_q E_q$ average of E.

V. CONCLUSION

In order to provide applications where quantum computers can practically outperform classical computers on hardware anticipated in the near-future, it is necessary to develop algorithms where there is a greater than

square-root speedup in the complexity [8]. This is because the large overheads involved in implementing quantum gates in an error-corrected code mean that there is a huge slowdown in the gate frequency as compared to classical computers. When the Betti number is of order 1, the complexity of the quantum algorithm for estimating Betti numbers is only a square root speedup over classical approaches. This is as compared to heuristic classical approaches that scale approximately linearly in $\binom{n}{k}$. On the other hand, when the Betti number is large, the complexity of estimating the Betti number to given *relative* error (error as a ratio to the Betti number) will be small.

There exist classes of graphs with very large Betti numbers. We introduce graph classes for a given k such that the speedup is a $k/2$ root, which is greater than a square-root speedup for $k > 4$, and can provide an arbitrarily large polynomial speedup for large k. These are very specially constructed graphs for large Betti number, but we show there exist far more general classes of graphs where a *quartic* speedup is possible, showing that speedups beyond quadratic are possible far more generally.

We have also provided a host of new techniques for quantum Betti number estimation that reduce the complexity. These include Kaiser-window amplitude estimation, improved Dicke state preparation, and improved eigenstate filtering. These improvements greatly improve the complexity in many ways, though the main scaling of the complexity as $\sqrt{\binom{n}{k}}$ remains. Our methods enable accurate estimation of the complexity of the quantum algorithm, including all constant factors.

Based on that, we estimate that tens of billions of Toffolis would be sufficient to estimate a Betti number that should be classically intractable. This number of Toffoli gates is reasonable for early generations of fully fault-tolerant quantum computers. While the exact threshold for quantum advantage depends on constant factors in the classical algorithms, it seems likely that this application will fall somewhere in between quantum chemistry applications and Shor's algorithm in terms of the resources required for quantum advantage. The standard classical approaches would be expected to be intractable because they would require an extremely large storage. We have also presented an alternative approach for classical estimation that could be more efficient, because it does not require large storage and instead requires Monte-Carlo sampling. That classical approach may be tractable, but it is difficult to evaluate its complexity because it depends on the gap of a Markov chain which is unknown.

There is scope for further improvement of the quantum algorithm for Betti numbers by implementing a more efficient method of clique finding. We have currently applied just amplitude amplification for clique finding, but there are more efficient classical methods for clique finding that could potentially be adapted for the quantum algorithm. That is nontrivial because these methods often require large storage, which would not be practical in the quantum algorithm where we need to minimize the number of ancilla qubits.

ACKNOWLEDGEMENTS

RB acknowledges helpful conversations with Seth Lloyd, who suggested to us that Erdős-Rényi graphs may have large Betti numbers. DWB acknowledges helpful discussions with Robin Kothari. JB acknowledges helpful conversations with David Gamarnik and Nikhil Srivastava. DWB worked on this project under a sponsored research agreement with Google Quantum AI. DWB is also supported by Australian Research Council Discovery Projects DP190102633 and DP210101367. NW and VD were funded by a grant from Google Quantum AI. NW was also funded by grants from the US Department of Energy, Office of Science, National Quantum Information Science Research Centers, Co-Design Center for Quantum Advantage under contract number DE-SC0012704. Some of the discussions and collaboration for this project occurred while using facilities at the Kavli Institute for Theoretical Physics, supported in part by the National Science Foundation under Grant No. NSF PHY-1748958. VD and CG were supported by the Dutch Research Council (NWO/ OCW), as part of the Quantum Software Consortium programme (project number 024.003.037).

[1] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi, "Quantum algorithms for topological and geometric analysis of data," Nature Communications **7**, 10138 (2016).

[2] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd, "Quantum machine learning," Nature **549**, 195–202 (2017).

[3] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost, "Quantum principal component analysis," Nature Physics **10**, 631–633 (2014).

[4] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd, "Quantum Algorithm for Linear Systems of Equations," Physical Review Letters **103**, 150502 (2009).

[5] Ewin Tang, "A quantum-inspired classical algorithm for recommendation systems," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (2019) pp. 217–228.

[6] Ewin Tang, "Quantum principal component analysis only achieves an exponential speedup because of its state preparation assumptions," Physical Review Letters **127**, 060503 (2021).

[7] Yuval R. Sanders, Dominic W. Berry, Pedro C.S. Costa, Louis W. Tessler, Nathan Wiebe, Craig Gidney, Hartmut Neven, and Ryan Babbush, "Compilation of fault-tolerant quantum heuristics for combinatorial optimization," PRX Quantum **1**, 020312 (2020).

[8] Ryan Babbush, Jarrod R. McClean, Michael Newman, Craig Gidney, Sergio Boixo, and Hartmut Neven, "Focus beyond quadratic speedups for error-corrected quantum advantage," PRX Quantum **2**, 010103 (2021).

[9] Sam Gunn and Niels Kornerup, "Review of a quantum algorithm for Betti numbers," arXiv:1906.07673 (2019).

[10] Shashanka Ubaru, Ismail Yunus Akhalwaya, Mark S. Squillante, Kenneth L. Clarkson, and Lior Horesh, "Quantum topological data analysis with linear depth and exponential speedup," arXiv:2108.02811 (2021).

[11] Ryu Hayakawa, "Quantum algorithm for persistent Betti numbers and topological data analysis," arXiv:2111.00433 (2021).

[12] Beno Eckmann, "Harmonische funktionen und randwertaufgaben in einem komplex," Commentarii mathematici Helvetici **17**, 240–255 (1944/45).

[13] J. Friedman, "Computing Betti numbers via combinatorial Laplacians," Algorithmica **21**, 331–346 (1998).

[14] Casper Gyurik, Chris Cade, and Vedran Dunjko, "Towards quantum advantage via topological data analysis," arXiv:2005.02607 (2020).

[15] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang, "Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning," in *Proceedings of the 52nd Annual ACM SIGACT symposium on theory of computing* (2020) pp. 387–400.

[16] Marcos Crichigno and Tamara Kohler, "Clique homology is QMA1-hard," arXiv:2209.11793 (2022).

[17] Michał Adamaszek and Juraj Stacho, "Complexity of simplicial homology and independence complexes of chordal graphs," Computational Geometry **57**, 8–18 (2016).

[18] Peter Scheiblechner, "On the complexity of deciding connectedness and computing Betti numbers of a complex algebraic variety," Journal of Complexity **23**, 359–379 (2007).

[19] Bojan Mohar, "Isoperimetric numbers of graphs," Journal of Combinatorial Theory, Series B **47**, 274–291 (1989).

[20] Anna Gundert and May Szedláky, "Higher dimensional discrete Cheeger inequalities," Journal of Computational Geometry **6**, 54 (2015).

[21] Joel Friedman, "Computing Betti numbers via combinatorial Laplacians," Algorithmica **21**, 331–346 (1998).

[22] Andreas Bärtschi and Stephan Eidenbenz, "Deterministic preparation of Dicke states," in *Fundamentals of Computation Theory*, edited by Leszek Antoni Gasieniec, Jesper Jansson, and Christos Levcopoulos (Springer International Publishing, Cham, 2019) pp. 126–139.

[23] Dominic W. Berry, Mária Kieferová, Artur Scherer, Yuval R. Sanders, Guang Hao Low, Nathan Wiebe, Craig Gidney, and Ryan Babbush, "Improved techniques for preparing eigenstates of fermionic Hamiltonians," npj Quantum Information **4**, 22 (2018).

[24] Craig Gidney and Martin Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," Quantum **5**, 433 (2021).

[25] Sara Ayman Metwalli, François Le Gall, and Rodney Van Meter, "Finding small and large k-clique instances on a quantum computer," IEEE Transactions on Quantum Engineering **1**, 1–11 (2020).

[26] Ian D. Kivlichan, Craig Gidney, Dominic W. Berry, Nathan Wiebe, Jarrod McClean, Wei Sun, Zhang Jiang, Nicholas Rubin, Austin Fowler, Alán Aspuru-Guzik, Hartmut Neven, and Ryan Babbush, "Improved Fault-Tolerant Quantum Simulation of Condensed-Phase Correlated Electrons via Trotterization," Quantum **4**, 296 (2020).

[27] Yuval R. Sanders, Dominic W. Berry, Pedro C. S. Costa, Louis W. Tessler, Nathan Wiebe, Craig Gidney, Hartmut Neven, and Ryan Babbush, "Compilation of Fault-Tolerant Quantum Heuristics for Combinatorial Optimization," PRX Quantum **1**, 020312–020382 (2020).

[28] Ryan Babbush, Craig Gidney, Dominic W. Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven, "Encoding electronic spectra in quantum circuits with linear T complexity," Phys. Rev. X **8**, 041015 (2018).

[29] Aram W. Harrow and Richard A. Low, "Random Quantum Circuits are Approximate 2-designs," Communications in Mathematical Physics **291**, 257–302 (2009).

[30] Shashanka Ubaru and Yousef Saad, "Fast methods for estimating the numerical rank of large matrices," in *International Conference on Machine Learning* (PMLR, 2016) pp. 468–477.

[31] Michał Adamaszek, "Extremal problems related to Betti numbers of flag complexes," Discrete Applied Mathematics **173**, 8–15 (2014).

[32] Joonho Lee, Dominic W. Berry, Craig Gidney, William J. Huggins, Jarrod R. McClean, Nathan Wiebe, and Ryan Babbush, "Even More Efficient Quantum Computations of Chemistry Through Tensor Hypercontraction," PRX Quantum **2**, 030305 (2021).
[33] Matthew Kahle, "Topology of random clique complexes," Discrete Mathematics **309**, 1658–1671 (2009).
[34] Michael Goff, "Extremal Betti numbers of Rips complexes," arXiv:0910.0040 (2009).
[35] Matthew Kahle, "Random geometric complexes," Discrete & Computational Geometry **45**, 553–573 (2011).
[36] John Lee, *Introduction to topological manifolds*, Vol. 202 (Springer Science & Business Media, 2010).
[37] Melissa R. McGuirl, "Homology and persistent homology bootcamp notes summer @ icerm 2017," (2017).
[38] Robert Ghrist, "Barcodes: the persistent topology of data," Bulletin of the American Mathematical Society **45**, 61–75 (2008).
[39] Vin Silva and Robert Ghrist, "Coverage in sensor networks via persistent homology," Algebraic and Geometric Topology **7** (2007), 10.2140/agt.2007.7.339.
[40] Afra Zomorodian and Gunnar Carlsson, "Computing persistent homology," Discrete & Computational Geometry **33**, 249–274 (2005).
[41] Herbert Edelsbrunner and John L Harer, *Computational topology: an introduction* (American Mathematical Society, 2022).
[42] Dominic W Berry, Graeme Ahokas, Richard Cleve, and Barry C Sanders, "Efficient quantum algorithms for simulating sparse Hamiltonians," Communications in Mathematical Physics **270**, 359–371 (2007).
[43] Olle Häggström, *Finite Markov chains and algorithmic applications* (Cambridge University Press, 2002).
[44] Andrew M. Childs, Yuan Su, Minh C. Tran, Nathan Wiebe, and Shuchen Zhu, "Theory of Trotter error with commutator scaling," Phys. Rev. X **11**, 011020 (2021).

Appendix A: Detailed background on topological data analysis

We now present background material from singular homology needed in topological data analysis, broadly following the treatments in [36–38].

Let $v_0, \ldots, v_k$ be $k+1$ distinct points in $\mathbb{R}^n$. The set $\{v_0, \ldots, v_k\}$ is said to be **affinely independent** if the set $\{v_1 - v_0, \ldots, v_k - v_0\}$ is linearly independent. In other words, we consider the given set of points to be affinely independent if when we take one of the points to be the "origin" (say v_0 WLOG) and draw vectors from this point to the others, the collection of the resulting vectors is linearly independent.

If $\{v_0, \ldots, v_k\}$ is affinely independent, the **k-simplex** spanned by them is the set

$$[v_0, \ldots, v_k] \coloneqq \left\{ \sum_{i=0}^{k} t_i v_i \colon t_i \geq 0 \text{ and } \sum_{i=0}^{k} t_i = 1 \right\} \tag{A1}$$

Equivalently, a simplex is just the convex hull of its affinely independent set of vertices. The points v_i are the **vertices** of the simplex and the integer k is the **dimension** of the simplex. Figure 4 shows some examples of simplices. Note that it follows from the definitions given that $k \leq n$ since any set of $n+2$ points or more cannot be affinely independent. This is because no collection of $n+1$ vectors or more in an n-dimensional vector space can be linearly independent. Such a collection of vectors therefore cannot determine any simplicies of dimension $n+1$ or higher.

Let σ be a k-simplex. A simplex spanned by a non-empty subset of the vertices of σ is a **face** of σ. For example, the 0-dimensional faces of σ are its vertices and its 1-dimensional faces are the edges, which are spanned by two vertices. Faces of σ that are not equal to σ are called **proper faces**. The $(k-1)$-dimensional faces of σ are called its **boundary faces** and their union is its **boundary**.

FIG. 4. A 0-simplex (point), 1-simplex (edge), 2-simplex (triangle), and 3-simplex (tetrahedron) from left to right.

Definition 2 (**(Euclidean) Simplicial Complex**). *A simplicial complex S is a finite collection of simplices satisfying the following conditions:*

1. *$\tau \subseteq \sigma$ for $\sigma \in S$ implies $\tau \in S$*

2. If $\sigma_1, \sigma_2 \in S$, then $\sigma_1 \cap \sigma_2 = \varnothing$ or $\sigma_1 \cap \sigma_2$ is a face of σ_1 and σ_2

The first condition says that a simplicial complex should also contain all the faces of a given simplex in the complex. The second condition says that any two simplices in a simplicial complex either do not intersect or intersect at a common face of both. We define the **dimension of a simplicial complex** to be the maximum of the dimensions of all simplices in the complex.

FIG. 5. (Left) A 2D simplicial complex in $\mathbb{R}^2$. (Right) A set that is not a simplicial complex in $\mathbb{R}^2$. It violates condition 2 of Definition 2.

It can be shown that a simplicial complex is completely determined by its vertices and information about which sets of vertices span which simplices. This provides the motivation for the following definition.

Definition 3 (**Abstract Simplicial Complex**). *An abstract simplicial complex is a collection C of non-empty finite sets such that if $s \in C$, then every non-empty subset of s is also in C.*

This general notion of a simplicial complex is particularly useful when we wish to construct one abstractly without reference to a particular embedding into Euclidean space.

Given a data set, we wish to create a simplicial complex from it and use this to approximate the topological properties of the underlying data. How well this approximation actually captures the underlying topological features of the data set is given by the Nerve Theorem.

Definition 4. *Let X be a topological space and let $C = \{U_i\}_{i \in I}$ be a cover of X, where I is some set of indices. The **nerve** $N(C)$ of C is the abstract simplicial complex with vertex set I, where $\{i_0, \ldots, i_k\} \subseteq I$ spans a k-simplex iff $U_{i_0} \cap \cdots U_{i_k} \neq \varnothing$.*

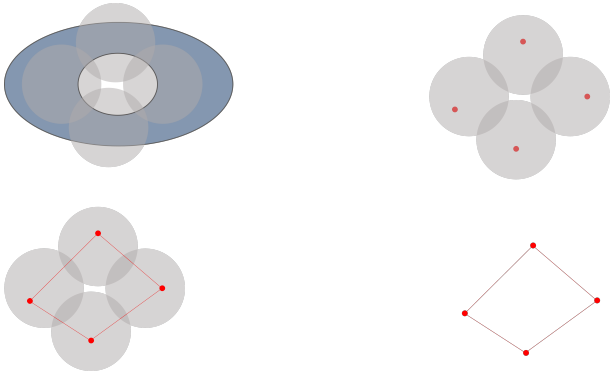

FIG. 6. (Top left) A cover of a metric space. (Top right) Vertex set of the cover in red. (Bottom left) Nerve of the cover superimposed over the cover of the metric space. (Bottom right) Nerve of the cover by itself. Source: Some random set of notes

Theorem 2. *(**Nerve Theorem**) Let C be a finite collection of closed, convex sets in Euclidean space. Then the nerve of C and the union of the sets in C have the same homotopy type.*

See Figure 6 for an example of the nerve of a cover. The Nerve Theorem implies that simplical complexes constructed out of a point cloud in Euclidean space have the same topological (more specifically homotopy invariant) properties as their covers by closed, convex sets (like closed balls in $\mathbb{R}^n$). This motivates the following simplicial complex construction in the case of points in $\mathbb{R}^n$:

Definition 5 (**Čech Complex**). *Let X be a collection of points in $\mathbb{R}^n$. For each $x \in X$, let $B_\epsilon(x)$ be a closed ball centered at x with radius ϵ. The Čech complex of X and ϵ is the nerve of $\{B_\epsilon(x)\}_{x\in X}$*

$$\mathcal{C}_\epsilon(X) = \{S \subset X \colon \cap_{x\in S} B_\epsilon(x) \neq \varnothing\} \tag{A2}$$

Therefore to construct a Čech complex out of a data set X, we draw ϵ neighborhoods around each point in our data set. Then if we want to determine whether a collection of $k+1$ points in X spans a k-simplex in the complex, we look at whether or not all the $k+1$ ϵ-neighborhoods around each point mutually intersect. Such a complex is assured to represent the same topological information as the cover of X by the ϵ-neighborhoods of its points from the Nerve Theorem.

Since we must determine if all ϵ-neighborhoods around each point in a collection of $k+1$ points intersect in order to create a k-simplex in a Čech complex, constructing this complex becomes computationally intractable for large data sets. It is therefore customary to instead construct the following simpler complex in practice:

Definition 6 (**Vietoris-Rips Complex**). *Let X be a collection of points in $\mathbb{R}^n$. The Vietoris-Rips complex of X and ϵ is*

$$\mathcal{R}_\epsilon(X) = \{S \subset X \colon B_\epsilon(x_i) \cap B_\epsilon(x_j) \neq \varnothing \ \forall x_i, x_j \in S\} \tag{A3}$$

Equivalently, the VR complex of a collection of points is the abstract simplicial complex whose k-simplices correspond to $(k+1)$-tuples of points which are pairwise within distance 2ϵ. This is computationally more feasible to determine compared to the Čech complex since the VR complex merely requires the computation of $O(n^2)$ distances, where n is the number of data points. Figure 7 shows the difference between the Rips and Čech complexes generated by a cover with an underlying vertex set.

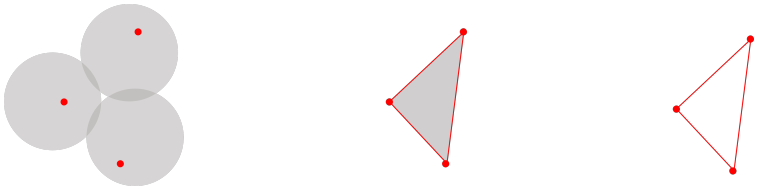

FIG. 7. (Left) A cover with underlying vertex set in red. (Middle) Rips complex generated from the cover. (Right) Čech complex generated from the cover.

The price paid for choosing VR complexes to construct is that the Nerve Theorem does not apply to them. Thus $\mathcal{R}_\epsilon(X)$ does not generally encode the same topological information as $\mathcal{C}_\epsilon(X)$ at a fixed ϵ and thus does not necessarily represent the actual topological features of the data set. Fortunately, the following proposition shows why VR complexes are nevertheless a good approximation to a Čech complex.

Proposition 5 (de Silva [39]). *For any $\epsilon > 0$, there is a chain of inclusion maps*

$$\mathcal{R}_\epsilon \hookrightarrow \mathcal{C}_{\epsilon\sqrt{2}} \hookrightarrow \mathcal{R}_{\epsilon\sqrt{2}}, \tag{A4}$$

where the simplicial complex is left implicit in the notation.

This proposition implies that any topological feature which persists under the inclusion $\mathcal{R}_\epsilon \hookrightarrow \mathcal{R}_{\epsilon'}$ is also a topological feature of $\mathcal{C}_{\epsilon'}$ if $\epsilon' \geq \sqrt{2}\epsilon$. Thus a topological feature (like a hole) in $\mathcal{R}_\epsilon$ that persists in $\mathcal{R}_{\epsilon\sqrt{2}}$ is also present in $\mathcal{C}_{\epsilon\sqrt{2}}$. This justifies the usage of a parameterized family of Rips complexes in extracting topological data.

The topological data we are most interested in are the Betti numbers of the simplicial complex $\mathcal{R}_\epsilon(X)$, which give the number of holes of a given dimension in that simplicial complex. We now show how to extract the Betti numbers of a simplicial complex via simplicial homology.

Let K be a simplicial complex. An **i-chain** is a formal sum of simplices $\sum_i c_i\sigma_i$ over all the i-simplices in K where $\sigma_i \in K$, $c_i \in \mathbb{F}$, and $\mathbb{F}$ is any field. The set of all i-chains is denoted by $C_i(K)$ and is a vector space over $\mathbb{F}$. The i-simplices form a basis for $C_i(K)$, so the dimension of $C_i(K)$ equals the number of i-simplices in K.

Definition 7 (**Boundary Map**). *Let $\sigma = [v_0, \ldots, v_k]$ be a k-simplex. The boundary map on k-simplices is a map*

$$\partial_k \colon C_k(X) \to C_{k-1}(X)$$

that acts as

$$\partial_k \sigma = \sum_{i=0}^{k} (-1)^i [u_0, u_1, \ldots, \hat{u}_i, \ldots, u_k]$$

where $\hat{u}_i$ denotes that the vertex i has been removed.

The boundary map acts on k-simplices $\sigma \in C_i(K)$ and gives a $(k-1)$-simplex $\partial_k \sigma$ that can be interpreted as the boundary of σ.

An **i-cycle** is an i-chain $c \in C_i(K)$ such that $\partial_i c = 0$. Therefore, i-cycles are precisely the kernel of the boundary map and are a subspace of $C_i(K)$ denoted by $Z_i = \ker \partial_i$. An i-chain c is an **i-boundary** if there exists an $(i+1)$-chain $\sigma \in C_{i+1}(K)$ such that $c = \partial_{i+1}(\sigma)$. Equivalently, i-boundaries are precisely the image of the boundary map and form a subspace denoted by $B_i(K) = \text{Im } \partial_{i+1}$. Figure 8 shows an example of a 1-boundary and 1-cycle.

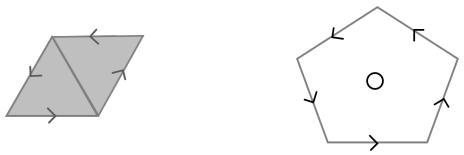

FIG. 8. (Left) The 1-boundary of a 2-chain as indicated by the arrows. (Right) A 1-cycle that is not the boundary of any 2-chain.

It turns out that there is a relationship between the two subspaces as implied by the following fundamental result in homology theory.

Proposition 6. *$\partial_i \circ \partial_{i+1}(\sigma) = 0$ for all $i+1$ chains σ and all $0 \leq i \leq \dim K$.*

The above proposition essentially says that the boundary of a boundary is 0. It implies that the image of ∂_{i+1} is contained in the kernel of ∂_i. This allows us to define the homology groups as follows:

Definition 8 (**Homology Groups**). *The i-th singular homology group H_i of a simplicial complex is the quotient group*

$$H_i(K) = Z_i(K)/B_i(K) = \text{Ker } \partial_i / \text{Im } \partial_{i+1}$$

Definition 9 (**Betti Numbers**). *The i-th Betti number β_i is the rank (dimension) of the i-th homology group.*

The i-th homology group is generated by cycles that are not the boundaries of any simplex. In other words, these are simplices that enclose a "void" or "hole" (see the right image of Figure 8 above). It is worth noting that β_0, the 0^{th} Betti number, represents the number of connected components the simplicial complex has.

The problem of computing the Betti numbers of a simplicial complex therefore reduces to the problem of computing the rank of the boundary map. A common and simple classical approach to doing this at the computational level is as follows.

Let K be a simplical complex and assume we are working over the field $\mathbb{Z}_2$. Label the p-simplices in $C_p(K)$ by $x_1, \ldots, x_{n_p}$ and the $(p-1)$-simplices in $C_{p-1}(K)$ by $y_1, \ldots, y_{n_{p-1}}$. These simplices form bases for $C_p(K)$ and $C_{p-1}(K)$ as mentioned previously. We can then represent the action of the boundary map ∂_p on $C_p(K)$ as follows

$$\partial_p(x_j) = \sum_{i=1}^{n_{p-1}} a_j^i y_i \text{ where } a_j^i = \begin{cases} 1 & \text{if } y_i \text{ is a face of } x_j \\ 0 & \text{otherwise} \end{cases} \tag{A5}$$

Then for any p-chain $c = \sum_{j=1}^{n_p} a_j x_j$, we can write the above in matrix form

$$\partial_p c = \begin{bmatrix} a_1^1 & a_1^2 & \dots & a_1^{n_p} \\ a_2^1 & a_2^2 & \dots & a_2^{n_p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_{p-1}}^1 & a_{n_{p-1}}^2 & \dots & a_{n_{p-1}}^{n_p} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{n_p} \end{bmatrix} \tag{A6}$$

Thus the boundary map on $C_p(K)$ can be represented as an $n_{p-1} \times n_p$ sparse matrix with entries in $\mathbb{Z}_2$. The columns of this matrix span $\text{Im } \partial_p = B_{p-1}$, so $\text{rank } \partial_p = \dim B_{p-1} = b_{p-1}$. The boundary matrix can brought into the Smith Normal Form via a generalization of Gaussian elimination which applies to any principal ideal domain (this includes fields). This results in a matrix with a number of 1's on the diagonal equal to the rank of the boundary matrix. Gaussian elimination for ∂_p takes $O(n_{p-1} n_p \min(n_p, n_{p-1}))$ time and requires $O((n_{p-1} + n_p)^2)$ memory [40, 41].

The full pipeline for doing persistence homology is as follows:

- Fix a starting value of a distance scale ϵ. This choice will be motivated by the nature of the data set
- Construct a VR complex from the data set at the scale ϵ
- Compute the Betti numbers of the VR complex up to a desired dimension (say, the dimension of the complex)
- Increase the value of epsilon by a desired amount and repeat the previous three steps
- Examine the Betti numbers obtained over the range of ϵ probed. The ones that "persist" over a wide range of ϵ indicate intrinsic topological features of the data set

1. Graphs

We will often be concerned with computing the Betti numbers of the clique complexes of graphs throughout this paper. The graphs serving as inputs into the quantum algorithms in this paper are not necessarily induced by any finite-dimensional data and are instead abstract graphs, i.e. abstract simpicial complexes in the sense of Definition 3 in the preceding section.

Definition 10 (**Graphs**). *A graph G is a pair of objects $G = (V, E)$, where V is a set of elements referred to as the "vertices" of G and E is a set consisting of pairs of vertices thought of as "edges" connecting the pairs of vertices.*

Given an undirected graph (i.e. one in which the edges are not assumed to have direction), a **clique** C of a graph is a subset of V such that every pair of distinct vertices in C is connected by an edge. C is called a k-clique if $|C| = k$.

We now define the notion of a clique complex:

Definition 11 (**Clique Complex**). *The clique complex of a graph G is the abstract simplical complex formed by associating a k-simplex to every $k+1$-clique in G.*

Note that this notion is analogous to the Vietoris-Rips complex defined previously in that clique complexes are determined by information about pairwise connectivity between vertices.

Appendix B: Alternative scheme for Dicke state preparation

Here we provide an alternative scheme for Dicke state preparation that makes more efficient use of ancilla qubits, but provides a lower amplitude for success. It is useful in the case of large n but small k (smaller than $\sqrt{n}$), where $\binom{n}{k} \sim n^k/k!$. The steps of this scheme are as follows.

1. First prepare k registers in equal superposition states over n values. The complexity of preparing an equal superposition over n values (with high probability of success) is $4\lceil \log n \rceil + 1$ Toffolis [7], so this would give complexity $k(4\lceil \log n \rceil + 1)$. In the case where n is a power of 2, then this preparation can be performed just with Hadamards.

2. For each of the k registers, apply unary iteration as in [28] with the n qubits for the Dicke state as the target. This is used to flip the corresponding qubit. The complexity of each unary iteration is $n - 2$, giving total complexity $k(n - 2)$. There will be approximately $\log n$ temporary ancillas used in the unary iteration that are reset to zero.

3. Now we sum the bits in the string of n qubits. A method of summing bits is given in [27], where multiple groups of bits are summed, and their sums are summed. The overall complexity is no more than $2n$ Toffolis, and only a logarithmic number of ancillas is used. The ancillas used that need to be kept for later stages of the calculation can be given as $\lceil \log n \rceil$, and there are $2\lceil \log n \rceil$ temporary ancillas used.

4. The sum of the bits is compared to k. This has complexity $\lceil \log k \rceil$ because we are guaranteed that the number of ones is at most k.

As before, this is a scheme which prepares the Dicke state in an entangled state with an ancilla. The overall complexity is then

$$(k + 2)n + k(4\lceil \log n \rceil - 1) + \lceil \log k \rceil \tag{B1}$$

Toffolis with the number of ancillas used being

$$(k + 1)\lceil \log n \rceil \tag{B2}$$

with a logarithmic number of working ancillas as well. In comparison, the scheme used above uses a number of qubits scaling as $n \log n$, so is much larger when $n \gg k$. In the case where n is a power of 2, the initial preparation of the equal superposition can be just performed with Hadamards, and so the Toffoli complexity is

$$(k + 2)n - 2k + \lceil \log k \rceil. \tag{B3}$$

There is a probability of failure for the preparation, because it is possible for the ones to overlap, which will be detected in the final stage where the sum of bits is compared to k. This is an example of the birthday problem, and the probability of success will be at least $1/2$ when k is not larger than approximately $\sqrt{n}$. The probability of success is more specifically given by

$$\frac{k!}{n^k} \binom{n}{k}. \tag{B4}$$

It is possible to perform amplitude amplification on this step, but it is simpler to combine this step with the clique finding. The net result on the complexity is that wherever we have $\binom{n}{k}$ in the original costs, it is replaced with $n^k/k!$.

Appendix C: Proof of complexity of amplitude estimation

For the complexity of amplitude estimation, the standard approach is to use phase estimation on the Grover iterate of amplitude amplification. If there is an initial amplitude of a, then the phase of each step of amplitude amplification is $2 \arcsin a$. The original proposal was to use control registers in the phase estimation in an equal superposition, but of course for phase estimation that is a poor choice. Here we would like a small probability of error beyond a given confidence interval, and for that case it is better to use a Kaiser window.

When applying phase measurement, we would start with a control state of the form (omitting normalisation)

$$\sum_{m=-N}^{N} \frac{1}{2N} \frac{I_0\left(\pi\alpha\sqrt{1 - (m/N)^2}\right)}{I_0(\pi\alpha)} |m\rangle. \tag{C1}$$

If we call the operator combining U and the reflection on the flag qubit W, then we would then control between applications of W and $W^\dagger$ with eigenvalue $e^{i\theta}$ to give

$$\sum_{m=-N}^{N} \frac{1}{2N} \frac{I_0\left(\pi\alpha\sqrt{1-(m/N)^2}\right)}{I_0(\pi\alpha)} e^{im\theta}|m\rangle. \tag{C2}$$

The inverse quantum Fourier transform then corresponds to an inner product with a phase state

$$\frac{1}{\sqrt{2N+1}} \sum_{m=-N}^{N} e^{im\hat{\theta}}|m\rangle. \tag{C3}$$

The inner product then gives the Fourier transform, so is proportional to

$$\frac{\sin\left(\sqrt{N^2(\hat{\theta}-\theta)^2-(\pi\alpha)^2}\right)}{I_0(\pi\alpha)\sqrt{N^2(\hat{\theta}-\theta)^2-(\pi\alpha)^2}}. \tag{C4}$$

This needs to be squared to give the probability distribution for the error in the phase measurement.

The distribution has its first zero for $\theta = (\pi/N)\sqrt{1+\alpha^2}$, so to estimate the probability in the wings of the distribution we should integrate past that point. We also have the difficulty that we are not given the exact normalisation of the probability distribution. To approximate the normalisation, we can approximate the centre of the distribution by a Gaussian. The approximation can be found by taking the Taylor series of the log of the distribution about zero, and gives

$$\frac{\sinh^2(\pi\alpha)}{\pi^2\alpha^2 I_0^2(\pi\alpha)} e^{-N^2(\pi\alpha\coth(\pi\alpha)-1)\Delta\theta^2/(\pi^2\alpha^2)}, \tag{C5}$$

where we have replaced $\hat{\theta}-\theta = \Delta\theta$. Taking the integral over $\Delta\theta$ then gives

$$\frac{\sinh^2(\pi\alpha)}{NI_0^2(\pi\alpha)\sqrt{\pi}\alpha\sqrt{\pi\alpha\coth(\pi\alpha)-1}}. \tag{C6}$$

It is found that this expression is asymptotically

$$\frac{\pi}{2N\sqrt{\alpha}} + \mathcal{O}(\alpha^{-3/2}). \tag{C7}$$

This can be found using the asymptotic properties of Bessel functions,

$$\frac{1}{I_0^2(\pi\alpha)} \approx \frac{2\pi^2\alpha}{e^{2\pi\alpha}} \tag{C8}$$

and $\sinh^2(\pi\alpha) \approx e^{2\pi\alpha}/4$ and $\coth(\pi\alpha) \approx 1$, so

$$\frac{\sinh^2(\pi\alpha)}{NI_0^2(\pi\alpha)\sqrt{\pi}\alpha\sqrt{\pi\alpha\coth(\pi\alpha)-1}} \approx \frac{2\pi^2\alpha}{e^{2\pi\alpha}} \frac{e^{2\pi\alpha}}{4} \frac{1}{N\sqrt{\pi}\alpha\sqrt{\pi\alpha-1}}. \tag{C9}$$

That gives the asymptotic expression claimed.

Now, for the integral over the tails we can upper bound the probability by that where we replace the sin with 1, so we have an upper bound

$$2\int_{(\pi/N)\sqrt{1+\alpha^2}}^{\infty} \frac{1}{I_0^2(\pi\alpha)[N^2\delta\theta^2-(\pi\alpha)^2]} d\Delta\theta = \frac{1}{I_0^2(\pi\alpha)} \frac{2\,\mathrm{arcsinh}(\alpha)}{\pi N\alpha}. \tag{C10}$$

Now using $\mathrm{arcsinh}(\alpha) \approx \ln(2\alpha)$, we have the asymptotic expression

$$\frac{2\pi^2\alpha}{e^{2\pi\alpha}} \frac{2\ln(2\alpha)}{\pi N\alpha} = \frac{4\pi\ln(2\alpha)}{Ne^{2\pi\alpha}}. \tag{C11}$$

Dividing by the asymptotic expression for the normalisation then gives

$$\frac{4\pi \ln(2\alpha)}{Ne^{2\pi\alpha}} \frac{2N\sqrt{\alpha}}{\pi} = 8\ln(2\alpha)\sqrt{\alpha}\, e^{-2\pi\alpha}. \tag{C12}$$

This tells us that, if we want probability of error outside the range given by δ, then we should take

$$\ln(1/\delta) \approx 2\pi\alpha - \ln[8\ln(2\alpha)\sqrt{\alpha}]. \tag{C13}$$

Solving for α then gives

$$\alpha = (1/2\pi)\ln(1/\delta) + \mathcal{O}(\ln\ln(1/\delta)). \tag{C14}$$

The size of the confidence interval is $(\pi/N)\sqrt{1+\alpha^2}$, so if that needs to be ϵ, we should take

$$N = \frac{\pi}{\epsilon}\sqrt{1+\alpha^2} = \frac{1}{2\epsilon}\ln(1/\delta) + \mathcal{O}(\epsilon^{-1}\ln\ln(1/\delta)). \tag{C15}$$

The higher-order $\ln\ln$ term for α is larger than the correction term for approximating $\sqrt{1+\alpha^2}$ by α. The number of calls to U or $U^\dagger$ is N, giving the complexity stated.

Appendix D: Qubitization with projection

Here we derive the expression for qubitization with a more general projection as given in Eq. (33). When the block encoding is defined more generally using

$$(|0\rangle\langle 0| \otimes P)\, V\, (|0\rangle\langle 0| \otimes P) = |0\rangle\langle 0| \otimes H/\lambda, \tag{D1}$$

then for $|k\rangle$ an eigenstate of H with energy E_k (and $P|k\rangle = |k\rangle$), we have

$$\begin{aligned}
(|0\rangle\langle 0| \otimes P)\, V\, (|0\rangle\langle 0| \otimes P)\, |0\rangle\,|k\rangle &= \frac{E_k}{\lambda}\,|0\rangle\,|k\rangle \\
(|0\rangle\langle 0| \otimes P)\, V\, |0\rangle\,|k\rangle &= \frac{E_k}{\lambda}\,|0\rangle\,|k\rangle \\
V\,|0\rangle\,|k\rangle &= \frac{E_k}{\lambda}\,|0\rangle\,|k\rangle + i\sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}\,\left|0k^\perp\right\rangle,
\end{aligned} \tag{D2}$$

where $\left|0k^\perp\right\rangle$ is defined as a state such that $(|0\rangle\langle 0| \otimes P)\left|0k^\perp\right\rangle = 0$. That is, V gives an application of H to the target system, with failure being flagged by states orthogonal to $|0\rangle$ on the ancilla or perpendicular to the projector P on the system. The phase factor on the orthogonal part can be chosen arbitrarily, and is chosen here as i to simplify later expressions. Then, one can define the qubiterate as $W := RV$ in the usual way except with the reflection being

$$R := i\,(2\,|0\rangle\langle 0| \otimes P - I)\,. \tag{D3}$$

This is similar to that in [23], except we have included the projection P in the reflection operation. Essentially the entire chain of reasoning as in [23] can be used, except replacing $|0\rangle\langle 0| \otimes I$ with $|0\rangle\langle 0| \otimes P$. Then we obtain

$$W\,|0\rangle\,|k\rangle = i\frac{E_k}{\lambda}\,|0\rangle\,|k\rangle + \sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}\,\left|0k^\perp\right\rangle. \tag{D4}$$

Then, to show the correct expression for $W\left|\chi k^\perp\right\rangle$ we use

$$\begin{aligned}
W^\dagger\,|\chi\rangle\,|k\rangle &= -VR\,|\chi\rangle\,|k\rangle \\
&= -iV\,|\chi\rangle\,|k\rangle
\end{aligned}$$

$$
\begin{aligned}
&= iRW \left|\chi\right\rangle \left|k\right\rangle \\
&= iR\left(i\frac{E_k}{\lambda}\left|\chi\right\rangle \left|k\right\rangle + \sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}\left|\chi k^\perp\right\rangle\right) \\
&= -i\frac{E_k}{\lambda}\left|\chi\right\rangle \left|k\right\rangle + \sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}\left|\chi k^\perp\right\rangle
\end{aligned} \tag{D5}
$$

Similarly, for $\left|\chi k^\perp\right\rangle$, we have

$$
\begin{aligned}
W^\dagger \left|\chi k^\perp\right\rangle &= -VR\left|\chi k^\perp\right\rangle \\
&= iV\left|\chi k^\perp\right\rangle
\end{aligned} \tag{D6}
$$

Now applying $W^\dagger$ to the expression for $W\left|\chi\right\rangle\left|k\right\rangle$ gives

$$
\begin{aligned}
\left|\chi\right\rangle\left|k\right\rangle &= i\frac{E_k}{\lambda}W^\dagger\left|\chi\right\rangle\left|k\right\rangle + \sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}W^\dagger\left|\chi k^\perp\right\rangle \\
&= i\frac{E_k}{\lambda}\left(-i\frac{E_k}{\lambda}\left|\chi\right\rangle\left|k\right\rangle + \sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}\left|\chi k^\perp\right\rangle\right) + i\sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}V\left|\chi k^\perp\right\rangle \\
&= \left|\frac{E_k}{\lambda}\right|^2\left|\chi\right\rangle\left|k\right\rangle + i\frac{E_k}{\lambda}\sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}\left|\chi k^\perp\right\rangle + i\sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}V\left|\chi k^\perp\right\rangle \\
\left(1-\left|\frac{E_k}{\lambda}\right|^2\right)\left|\chi\right\rangle\left|k\right\rangle &= i\frac{E_k}{\lambda}\sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}\left|\chi k^\perp\right\rangle + i\sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}V\left|\chi k^\perp\right\rangle \\
-i\sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}\left|\chi\right\rangle\left|k\right\rangle &= \frac{E_k}{\lambda}\left|\chi k^\perp\right\rangle + V\left|\chi k^\perp\right\rangle \\
V\left|\chi k^\perp\right\rangle &= -\frac{E_k}{\lambda}\left|\chi k^\perp\right\rangle - i\sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}\left|\chi\right\rangle\left|k\right\rangle \\
RV\left|\chi k^\perp\right\rangle &= i\frac{E_k}{\lambda}\left|\chi k^\perp\right\rangle + \sqrt{1-\left|\frac{E_k}{\lambda}\right|^2}\left|\chi\right\rangle\left|k\right\rangle
\end{aligned} \tag{D7}
$$

Hence we obtain Eq. (33) as required. We have corrected the extra factors of i included in [23]. Note that the rest of the reasoning in [23] is correct.

Appendix E: Betti number and spectral gap calculations

The purpose of this section is to prove Propositions 1, 2, 3 and 4.

Definition 12. *Given two simplicial complexes X and Y, define their* join $X * Y$ *to be the simplicial complex consisting of faces $\sigma \otimes \tau := \sigma \cup \tau$ for all $\sigma \in X$, $\tau \in Y$.*

Observe that $K(m,k) = K(m,k-1) * K(m,1)$.

In this section, we will work with *reduced* homology, denoted by a tilde. This is identical to regular homology, except that we have an extra 1-dimensional space C_{-1} and an extra boundary map $\partial_0 : C_0 \to C_{-1}$ which maps every vertex (0-simplex) to the unique basis vector of C_{-1}. This has the effect that the reduced homology $\widetilde{H}_0$ is equal to the number of connected components *minus one*, rather than simply the number of connected components. The rest of the homology groups are unchanged $\widetilde{H}_k = H_k,\ k > 0$.

The homology of the join is given by the well-known Kunneth formula.

Lemma 2. (Kunneth formula)

$$\widetilde{H}_k(X * Y) = \bigoplus_{i+j=k-1} \widetilde{H}_i(X) \otimes \widetilde{H}_j(Y) \tag{E1}$$

$$\implies \widetilde{\beta}_k(X * Y) = \sum_{i+j=k-1} \widetilde{\beta}_i(X)\widetilde{\beta}_j(Y) \tag{E2}$$

We would also like to relate the Laplacian of $X * Y$ to the Laplacians of X and Y.

Lemma 3. *Let $\sigma \in X$ be an i-simplex and $\tau \in Y$ a j-simplex with $i + j = k - 1$. Then*

$$\Delta_k^{X*Y}(\sigma \otimes \tau) = (\Delta_i^X \sigma) \otimes \tau + \sigma \otimes (\Delta_j^Y \tau) \tag{E3}$$

Proof. Let's work in the *graded algebra* $C_{-1} \oplus C_0 \oplus C_1 \oplus \dots$. We have

$$\begin{aligned}
\Delta &= \partial^\dagger \partial + \partial \partial^\dagger \\
\partial(\sigma \otimes \tau) &= (\partial\sigma) \otimes \tau + (-1)^{|\sigma|}\sigma \otimes (\partial\tau) \\
\partial^\dagger(\sigma \otimes \tau) &= (\partial^\dagger\sigma) \otimes \tau + (-1)^{|\sigma|}\sigma \otimes (\partial^\dagger\tau) \\
\implies \Delta(\sigma \otimes \tau) &= (\Delta\sigma) \otimes \tau + \sigma \otimes (\Delta\tau)
\end{aligned}$$

□

Corollary 1. *Let* $\operatorname{spec}\Delta$ *denote the set of eigenvalues of* Δ.

$$\operatorname{spec}\Delta_k^{X*Y} = \bigcup_{i+j=k-1} \operatorname{spec}\Delta_i^X + \operatorname{spec}\Delta_j^Y \tag{E4}$$

Here the plus notation for sets means $A + B = \{a + b : a \in A,\ b \in B\}$.

Proof. Use Lemma 3 and let $\sigma \in C_i^X$ and $\tau \in C_j^Y$ be eigenchains of Δ_i^X and Δ_j^Y respectively. □

Proposition 7. (Restatement of Proposition 1.)
The $(k-1)^{th}$ Betti number of (the clique complex of) $K(m,k)$ is

$$\beta_{k-1} = (m-1)^k \tag{E5}$$

Proof. $K(m,k) = K(m,k-1) * K(m,1)$ and the Betti numbers of $K(m,1)$ are $(m-1, 0, 0, \dots)$. Thus by induction using the Kunneth formula, we have $\beta_{k-1} = (m-1)^k$. □

Proposition 8. (Restatement of Proposition 2.)
The combinatorial Laplacian $\Delta_{k-1} = \partial_{k-1}^\dagger \partial_{k-1} + \partial_k \partial_k^\dagger$ of (the clique complex of) $K(m,k)$ has spectral gap

$$\lambda_{\min} = m \tag{E6}$$

Proof. Again $K(m,k) = K(m,k-1) * K(m,1)$. The spectrum of the $\Delta_0^{K(m,1)}$ is 0 with multiplicity $m-1$, and m with multiplicity 1. Thus by induction using Corollary 1, the spectrum of $\Delta_{k-1}^{K(m,k)}$ is (ignoring multiplicities) $\{0, m, 2m, \dots, km\}$. This gives $\lambda_{\min} = m$. □

Proposition 9. (Restatement of Proposition 3.)
The Rips complex described in Section IV C has $(2k-1)^{th}$ Betti number given by

$$\beta_{2k-1}(S) = (m-1)^k = \left(\frac{n}{2k} - 1\right)^k$$

Proof. The Betti numbers of $\mathcal{R}_1(S_i)$ are $(0, m-1, 0, \dots)$. Furthermore, $R_1(S) = R_1(S_0) * \dots * R_1(S_{k-1})$. Thus by the Kunneth formula $\beta_{2k-1}(S) = (m-1)^k$. □

Proposition 10. (Restatement of Proposition 4.)
The Rips complex described in Section IV C has a combinatorial Laplacian $\Delta_k = \partial_k^\dagger \partial_k + \partial_{k+1}\partial_{k+1}^\dagger$ with a constant spectral gap $\lambda_{\min}$.

Proof. Consider first the Laplacian $\Delta_k^{S_i}$ of $\mathcal{R}_1(S_i)$. Say it has smallest eigenvalue c. Then by induction using Corollary 1, the smallest eigenvalue of Δ_k^S is also c. □

Appendix F: Dequantization using path integral Monte Carlo

Previously, we argued that the cases where TDA can potentially have a super-polynomial advantage relative to classical approaches is in cases where the clique density is high. In such cases sampling in our quantum algorithm is efficient and eigen-decomposition is inefficient. However, we will see here that this is not necessarily the case and that there are cases where the clique density is high wherein randomized classical algorithms can achieve scaling that is polynomially equivalent to quantum algorithms. This will show that the conditions for a substantial improvement using quantum TDA are potentially even more subtle than previous work suggests.

First note that ∂_k as defined in Appendix A acts on the vector space of k-simplices with dimension $d_k = \binom{n}{k+1}$ and has column sparsity equal to $k+1$. The latter property is because ∂_k has non-zero elements only when an $k-1$-simplex is the boundary of a given k-simplex. Since there are $\binom{k+1}{k} = k+1$ simplices of dimension $k-1$ in the boundary of an k-simplex, the boundary map $k+1$ column-sparse. The row sparsity however is upper bounded by the maximal degree d of a vertex in the graph or by $(n-k-1)$ depending on which quantity is smaller. Moreover, the column and row sparsity of $\Delta_{k-1} = \partial_{k-1}^\dagger \partial_{k-1} + \partial_k \partial_k^\dagger$ is equal to the *degree of a k-simplex* c, i.e. the number of k-simplices c' such that $c \cap c'$ is a $(k-1)$-simplex and $c \cup c'$ is a $(k+1)$-simplex. A trivial upper bound on the degree of a k-simplex is $(k+1)(n-k-1)$; this is because in order to construct for a given c a c' as above, we can remove any of the $k+1$ vertices from c and add any of the $(n-k-1)$ vertices not in c to form c'. However, tighter bounds can be achieved by also using the largest degree d of a vertex, e.g. another upper bound on the sparsity is $O(kd)$.

As discussed in the main body, our dequantization looks at imaginary time simulations of the Hermitian operator $\widetilde{B_G^2} = B_G^2 + (1-P)\gamma_{\min}$, where B_G is the square of the constrained Dirac operator and the projector selects all input states that are valid simplices. Using the upper bound on the sparsity $O(kd)$ mentioned above, we can obtain the decomposition $\widetilde{B_G^2} = \sum_{p=1}^{D} c_p H_p$, where H_p is one-sparse and unitary and $D = O(k^2 d^2)$, in polynomial time using distributed graph coloring algorithms. These algorithms also let us compute the position of the non-zero matrix element in row x of U_α using a number of queries to Δ_k that scales as $O(\log^*(D))$ [42].

With the worst case scaling of D discussed, we can now proceed with our discussion of how to set up the path integral approximation to the normalized Betti number. In order to set up the path integrals, we must first employ a Trotter-decomposition. This allows us to represent the exponential in terms of exponentials of the one-sparse matrices, which can then be simulated through randomization. This leads us to the conclusion that

$$e^{-\widetilde{B_G^2} t} = (e^{-\widetilde{B_G^2} t/r})^r = \left(\prod_{p=1}^{D} e^{-c_p H_p t/2r} \prod_{p=D}^{1} e^{-c_p H_p t/2r} + O\left(\frac{\left(\sum_p |c_p|\right)^3 t^3}{r^3} \right) \right)^r . \tag{F1}$$

As H_p is Hermitian, it has a complete set of eigenvectors. H_p is also one-sparse and since such matrices can be written as the direct sum of irreducible one and two-dimensional matrices, we can parameterize the eigenvectors to respect the structure of the two dimensional space via

$$H_p \left|\lambda_{p,\nu}\right\rangle = \lambda_{p,\nu} \left|\lambda_{p,\nu}\right\rangle \tag{F2}$$

Note that each eigenvector $|\lambda_{p,\nu}\rangle$ is such that $\langle p|\lambda_{p,\nu}\rangle$ is non-zero for only two different computational basis vectors.

Note that the first term on the RHS of (F1) contains $2rD$ terms. If we introduce a vector of indices $p = \{1,\ldots,r,1,\ldots,r,\ldots,r\}$ with $2rD$ entries denoted by p_i, we can express the expectation of the exponential of the boundary operator as (H stands for the Haar average)

$$\mathbb{E}^H_{|\psi\rangle} \left(\prod_{i=1}^{2rD} e^{-c_{p_i} H_{p_i} t/2r} \right) := \mathbb{E}_H \left\langle \psi \right| \left(\prod_{i=1}^{2rD} e^{-c_{p_i} H_{p_i} t/2r} \right) \left| \psi \right\rangle .$$

Next we set up our path integrals by selecting sets of $2rD$ indices that correspond to the eigenstates that we transition to in the path integral. We denote such a path via the vector Γ where Γ_j corresponds to the index of the j^{th} eigenstate in the path. Using this notation we can insert resolutions of the identity of the

form $\sum_{\Gamma_j} \left|\lambda_{p_j,\Gamma_j}\right\rangle \left\langle\lambda_{p_j,\Gamma_j}\right|$ consisting of the eigenvectors $\left|\lambda_{p_j,\Gamma_j}\right\rangle$ of each U_{p_j} in between each of the $2rD$ terms and defining $\rho = |\psi\rangle\langle\psi|$ gives

$$\begin{aligned}
&\mathbb{E}^H_{|\psi\rangle}\left(\prod_{i=1}^{2rD} e^{-c_{p_i}H_{p_i}t/2r}\right) = \mathbb{E}_H \ \mathrm{Tr}\left(\rho \prod_{i=1}^{2rD} e^{-c_{p_i}H_{p_i}t/2r}\right) \\
&= \mathbb{E}_H \mathrm{Tr}\left(\rho \sum_{\Gamma_1,\ldots,\Gamma_{2rD}} \exp\left(-\sum_{i=1} \lambda_{p_i,\Gamma_i} t/2r\right) |\lambda_{p_1,\Gamma_1}\rangle\langle\lambda_{p_1,\Gamma_1}| \cdots |\lambda_{p_{2rD},\Gamma_{2rD}}\rangle\langle\lambda_{p_{2rD},\Gamma_{2rD}}|\right) \\
&= \mathbb{E}_H \mathrm{Tr}\left(\rho \sum_{\Gamma_1,\ldots,\Gamma_{2rD}} \exp\left(-\sum_{i=1} \lambda_{p_i,\Gamma_i} t/2r\right) W(\Gamma) |\lambda_{p_1,\Gamma_1}\rangle\langle\lambda_{p_{2rD},\Gamma_{2rD}}|\right) \\
&= \mathbb{E}_H \mathrm{Tr}\left(\rho \ \mathbb{E}_\Gamma \frac{\exp\left(-\sum_{i=1} \lambda_{p_i,\Gamma_i} t/2r\right) W(\Gamma) |\lambda_{p_1,k_1}\rangle\langle\lambda_{p_{2rD},k_{2rD}}|}{\Pr(\Gamma)}\right) \\
&= \mathbb{E}_H \mathbb{E}_\Gamma \mathrm{Tr}\left(\rho \frac{\exp\left(-\sum_{i=1} \lambda_{p_i,\Gamma_i} t/2r\right) W(\Gamma) |\lambda_{p_1,\Gamma_1}\rangle\langle\lambda_{p_{2rD},\Gamma_{2rD}}|}{\Pr(\Gamma)}\right) \\
&= \frac{1}{d_k} \mathbb{E}_\Gamma \mathrm{Tr}\left(\frac{\exp\left(-\sum_{i=1} \lambda_{p_i,\Gamma_i} t/2r\right) W(\Gamma) |\lambda_{p_1,\Gamma_1}\rangle\langle\lambda_{p_{2rD},\Gamma_{2rD}}|}{\Pr(\Gamma)}\right) \\
&= \frac{1}{d_k} \mathbb{E}_\Gamma \left(\frac{\exp\left(-\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/2r\right) W(\Gamma) \delta_{\Gamma_1,\Gamma_{2rD}}}{\Pr(\Gamma)}\right)
\end{aligned} \tag{F3}$$

where we have defined for convenience the quantity

$$W(\Gamma) = \langle\lambda_{p_1,\Gamma_1}|\lambda_{p_2,\Gamma_2}\rangle \cdots \left\langle\lambda_{p_{2rD-1},\Gamma_{2rD-1}}\middle|\lambda_{p_{2rD},\Gamma_{2rD}}\right\rangle \tag{F4}$$

with $\Gamma = [\Gamma_1, \ldots, \Gamma_{2rD}]$. In the fourth line, we divided and multiplied by a probability $\Pr(\Gamma)$ to express the sum as an average, which allows to use importance sampling to minimize the variance via a judicious choice of $\Pr(\Gamma)$. In the last line, we used the fact that $p_{2rD} = p_1$ for the symmetric Trotter formula.

If we wish to estimate this value by sampling, the primary driver of the complexity will be the estimation of the expectation value through the sample mean which corresponds to the optimal unbiased estimator of the population mean. The number of samples scales with the variance of the set that one averages over and the variance over Γ of the above Haar expectation is then simply

$$\begin{aligned}
&\mathbb{V}_\Gamma\left(\frac{1}{d_k}\mathrm{Tr}\left(\frac{\exp\left(-\sum_{i=1} \lambda_{p_i,\Gamma_i} t/2r\right) W(\Gamma) |\lambda_{p_1,\Gamma_1}\rangle\langle\lambda_{p_{2rD},\Gamma_{2rD}}|}{\Pr(\Gamma)}\right)\right) \\
&= \frac{1}{d_k^2} \sum_{\Gamma_1,\ldots,\Gamma_{2rD-1}} \frac{\exp\left(-2\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/r\right) |W(\Gamma)|^2 \delta_{\Gamma_1,\Gamma_{2rD}}}{\Pr(\Gamma)} \\
&\quad - \left(\frac{1}{d_k}\mathbb{E}_\Gamma\left(\frac{\exp\left(-\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/2r\right) W(\Gamma) \delta_{\Gamma_1,\Gamma_{2rD}}}{\Pr(\Gamma)}\right)\right)^2 \\
&\leq \frac{1}{d_k^2} \sum_{\Gamma_1,\ldots,\Gamma_{2rD-1}} \frac{\exp\left(-2\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/r\right) |W(\Gamma)|^2 \delta_{\Gamma_1,\Gamma_{2rD}}}{\Pr(\Gamma)}
\end{aligned} \tag{F5}$$

There are many probability distributions that we could choose to sample from to minimize the variance in (F5). The most straight forward distribution to choose, and the appropriate one to pick in the limit of short t, is a uniform distribution. However, in practice the eigenvalues in the sum may have wildly varying sizes and so the importance of each of the different paths can swing substantially. A more natural choice to make for the probability of drawing each path is

$$\Pr(\Gamma) = \frac{\exp\left(-2\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/r\right) \delta_{\Gamma \in S_\Gamma}}{\sum_{\Gamma \in S_\Gamma} \exp\left(-2\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/r\right)}, \tag{F6}$$

where S_Γ is the set of valid paths with $2rD$ vertices such that each edge corresponds to a path of connected eigenvectors for the one-sparse matrices used in the decomposition of $\widetilde{B_G^2}$.

The central challenge in employing this formula is to estimate the value of the sums over the values of Γ_i. Let us further assume that the H_j in used in the derivation are Hermitian and unitary such that

$$\lambda_{p_i,\Gamma_i} = \pm c_{p_i}. \tag{F7}$$

Furthermore, a one-sparse decomposition is chosen the matrix elements are all off diagonal with the exception of any diagonal matrix that appears in the decomposition. This can be seen explicitly using the discussion of the Jordan-Wigner representation of the Dirac operator. This means that each eigenvector couples to at most two eigenvectors. At most one term is diagonal in the standard Trotter decomposition of the Dirac operator [42]. A simple combinatorial argument leads to the conclusion that the total number of valid paths is at most $d_k 2^{2r(D-1)-1}$.

Given this choice, the normalization constant (which is analogous to a partition function) can be expressed (assuming that the diagonal element is always $p_i = D$) as

$$\begin{aligned}
&\sum_{\Gamma\in G} \exp\left(-2\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/r\right) \\
&\quad = d_k 2^{2rD-1-r} \cosh(2c_{p_1} t/r) \prod_{i=2}^{2rD-1} (\cosh(c_{p_i} t/r)\delta_{p_i\neq D} + \delta_{p_i=D} e^{-\lambda_{p_i,\Gamma_i} t/r}/2).
\end{aligned} \tag{F8}$$

This can be computed using $O(\text{poly}(rD))$ arithmetic operations and so does not ruin the efficiency of the algorithm. Note that were the sum over the $W(\Gamma)$ terms considered instead, then the result would be computationally difficult to compute as these terms generate correlations that would prevent us from performing an independent sum for each of the factors.

The variance σ^2 over the values of k chosen in the path integrals for the expression for the Haar average is then

$$\begin{aligned}
\sigma^2 &= \frac{1}{d_k^2}\left(\sum_{\Gamma\in S_\Gamma} |W(\Gamma)|^2\right)\left(\sum_{\Gamma\in S_\Gamma} \exp\left(-2\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/r\right)\right) \\
&\quad - \frac{1}{d_k^2}\left|\sum_{\Gamma\in S_\Gamma} \exp\left(-\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/2r\right) W(\Gamma)\right|^2 \\
&\leq \frac{2^{2rD} e^{2Dt \max_i \lambda_{p_i,\Gamma_i}}}{d_k} \\
&\leq \frac{2^{2rD} (1/\delta)^{2D \max_i |\lambda_{p_i,\Gamma_i}|/\gamma_{\min}}}{d_k}
\end{aligned} \tag{F9}$$

This shows that the variance after making this substitution is precisely equal to the gap in the Cauchy-Schwarz inequalty in the latter sum. This suggests that in certain cases where the Cauchy-Schwarz inequality is tight, the variance may be extremely small given that we have the ability to sample from the distribution $\Pr(\Gamma)$.

1. Metropolis Hastings algorithm

Outside of specific cases such as graphs, it is difficult in general to sample from the probability distribution $\Pr(\Gamma)$ to employ the above variance reduction strategy. It is therefore necessary to provide a general method to obtain these samples if we wish to understand how we could address the problem more generally. One way to address the issue of how to sample from the distribution $\Pr(\Gamma)$ is to use the Metropolis Hastings algorithm. The idea behind the algorithm is to design a Markov chain whose stationary distribution equals our choice of $\Pr(\Gamma)$.

We first start with a connected, undirected graph on the set of all possible states $\Gamma_1, \Gamma_2, \ldots, \Gamma_{2rD-1}$ which represent the "paths" involved in our Trotter decomposition. Each vertex of the graph then represents one possible collection of values for $\Gamma_1, \ldots, \Gamma_{2rD-1}$.

At each vertex a, we therefore select a neighbor with probability $1/(2rD-1)$. Since the degree may be less than $2rD-1$ at a given vertex, the walk may remain at that vertex as there is a non-zero probability of no edge being selected. To account for such situations, we have the following rules: if a neighboring vertex b is selected and the probability of transitioning

$$p_b \coloneqq \Pr(\Gamma)_b = \exp\left(-2\lambda_{p_1,\Gamma_1^{(b)}} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i^{(b)}} t/r\right)$$

where $b = \{\Gamma_i^{(b)} : i = 1 \dots 2rD-1\}$, is at least as great as the probability of remaining $p_a \coloneqq \Pr(\Gamma)_a$, we transition to b. If $p_b < p_a$, then we transition to b with probability p_b/p_a, where $b \in \mathcal{N}(a)$ with $\mathcal{N}(a)$ referring to the neighbors of a and

$$\frac{p_b}{p_a} = \frac{\exp\left(-2\lambda_{p_1,\Gamma_1^{(b)}} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i^{(b)}} t/r\right)}{\exp\left(-2\lambda_{p_1,\Gamma_1^{(a)}} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i^{(a)}} t/r\right)}. \tag{F10}$$

Otherwise we remain at a with probability $1 - p_b/p_a$. Defining

$$p_{ab} \coloneqq \frac{1}{R} \min\left(1, \frac{p_b}{p_a}\right)$$

and

$$p_{aa} \coloneqq 1 - \sum_{b \neq a} p_{ab}$$

we can easily verify $p_a p_{ab} = p_b p_{ba}$. By the Fundamental Theorem of Markov Chains [43], it follows that the stationary probabilities are p_a as needed.

The cost of computing the ratio p_b/p_a is $O(rD)$ arithmetic operations, which coincides with the cost of performing an update. The number of such updates needed to reach δ error from the stationary distribution, where δ is the total variational distance (TVD) from the stationary distribution desired, is

$$T^* \in O\left(\frac{\log(1/\delta)}{\gamma_M}\right), \tag{F11}$$

where $\gamma_M \coloneqq 1 - \lambda_2$ is the eigenvalue gap of for the transition matrix p. This implies that the number of arithmetic operations needed to sample from a distribution that is $\delta-$close to the stationary distribution is in

$$O\left(\frac{rD\log(1/\delta)}{\gamma_M}\right). \tag{F12}$$

If one samples from a distribution, P', that is $\delta-$close to the intended distribution P then the expectation value of any function f is $|\sum_j (P(j)f(j)) - \sum_j (P'(j)f(j))| \leq \delta \max|f(j)|$. Similarly, the variance obeys $|\sum_j P'(j)f(j)^2 - (\sum_j P'(j)f(j))^2| \leq \mathbb{V}(f) + O(\delta \max|f(j)|^2)$. Thus if we want the error in the mean to be less than some error ϵ_M, we require

$$\delta = \epsilon_M / \max|f(j)|.$$

Note that in our situation

$$|f| = \left|\frac{\exp\left(-\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/2r\right) W(\Gamma)}{\Pr(\Gamma)}\right|$$

where r is the number time-steps needed in the Trotterization procedure to attain a desired Trotter error ϵ_T. Further, we have the following bound

$$\sum_{\Gamma_1,\dots,\Gamma_{2rD-1}} \left|W(\Gamma) \exp\left(-2\lambda_{p_1,\Gamma_1} t/r - \sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/r\right)\right|$$

$$\leq \sqrt{\sum_{\Gamma_1,\ldots,\Gamma_{2rD-1}} |W(\Gamma)|^2} \sqrt{\sum_{\Gamma\in S_\Gamma} \exp\left(-4\lambda_{p_1,\Gamma_1} t/r - 2\sum_{i=2}^{2rD-1} \lambda_{p_i,\Gamma_i} t/r\right)} \tag{F13}$$

Note that when we sum over a specific p_i, Γ_i in $|W(\Gamma)|^2$, we get 1 since the eigenvectors are orthonormal. Performing all $2rD-2$ sums therefore gives 1 from all the inner products and the last sum involving the $(2rD-1)$-th index gives a factor of d_k. Additionally, each λ_{p_i,Γ_i} can either be positive or negative but is upper-bounded by $\|\widetilde{B_G^2}\|_{\max} \leq \|\widetilde{B_G^2}\|_\infty = \gamma_{\max}$, i.e. the largest eigenvalue of $\widetilde{B_G^2}$. We can then bound $f_{\max}$ as follows

$$|f| \leq f_{\max} \leq d_k e^{2\gamma_{\max} tD} 2^{r(D-1/2)}. \tag{F14}$$

2. Trotter error in path integration

From Corollary 12 of [44], the multiplicative Trotter error m for a p-th order Trotter formula is asymptotically bounded by

$$O\left(\alpha \left(\frac{t}{r}\right)^{p+1} \exp\left(\frac{2t}{r}\Upsilon \sum_{\ell=1}^{\Gamma} \|H_\ell\|\right)\right)$$

where the operator H is decomposed into a sum of Γ terms,

$$\alpha = \sum_{\ell_1,\ell_2,\ldots,\ell_{p+1}=1} \|[H_{\ell_{p+1}},\ldots,[H_{\ell_2},H_{\ell_1}]\cdots]\|$$

and Υ is the number of "stages" of the formula. For the symmetric Trotter-Suzuki formula, $\Upsilon = 2(5)^{q-1}$ for a TS formula of order $2q$, where $q=1$ and $p=2$ for our case. $\Gamma = 2rD$ and α can be upper bounded by

$$\alpha \leq (2rD)4 \max_p (c_p)^3 \leq 8rD\gamma_{\max}^3 \tag{F15}$$

Since this is the short time multiplicative error bound for simulating an operator for time t/r, we want to bound the resulting error when simulating for large t. To this end note that if we have an operator A we approximate by an operator B up to some multiplicative error m, then $B = A(I+mC)$ where C is an operator such that $\|C\| \leq 1$, and m is a constant. Then

$$\begin{aligned}
\|B\|^r &\leq (\|A\|(I+m\|C\|)^r \leq \|A\|^r \left(1 + \sum_{q=1}^{r} (m\|C\|)^q \binom{r}{q}\right) \\
&\leq \|A\|^r \left(1 + \sum_{q=1}^{r} m^q \binom{r}{q}\right) \leq \|A\|^r \left(1 + \sum_{q=1}^{r} \left(\frac{mre}{q}\right)^q\right) \\
&\leq \|A\|^r \left(1 + \sum_{q=1}^{r} (mre)^q\right) \leq \|A\|^r \left(1 + \frac{mre}{1-mre}\right)
\end{aligned}$$

Therefore the long-time multiplicative error is bounded by $mre/(1-mre)$ and we would like this to be less than some desired error $\epsilon_T > 0$. This implies that we must have $mre \leq \epsilon_T/(1+\epsilon_T) \leq \epsilon_T$. In our context, $A = e^{tH/r}$, B is an approximation to A as given by a Trotter formula, and m is the short-time multiplicative Trotter error bound cited above.

Using the bound on m and substituting in the parameters relevant for our situation, we have

$$mre \leq e\alpha \frac{t^3}{r^2} \exp\left(\frac{4t}{r} \sum_{\ell=1}^{2rD} \|H_\ell\|\right) \leq \epsilon_T$$

If $r \geq \frac{4t}{\ln 2}\sum_\ell \|H_\ell\|$, then

$$mre \leq 2e\alpha\frac{t^3}{r} \leq 2e\alpha\frac{t^3}{r^2} \leq \epsilon_T$$

which implies that

$$r = t\max\left\{\left(\frac{4et\alpha}{\epsilon_T}\right)^{1/2}, \frac{4}{\ln 2}\sum_\ell \|H_\ell\|\right\} \tag{F16}$$

The former term dominates asymptotically, so we will henceforth take $r \in \Theta\left(t\left(\frac{4et\alpha}{\epsilon_T}\right)^{1/2}\right)$.

The systematic error in the estimate of the expectation value from the Trotter-Suzuki formula and the finite length Markov chain is at most

$$\|e^{-\widetilde{B_G^2}t}\|\epsilon_T + \epsilon_M := \epsilon_{TM} \tag{F17}$$

The total number of operations needed to draw a single sample from the distribution with bias at most ϵ_{TM} is from (F12) in

$$O\left(\frac{rD\log(1/\delta)}{\gamma_M}\right) = O\left(\frac{\sqrt{\alpha}t^{3/2}D\log(f_{\max}/\epsilon_M)}{\sqrt{\epsilon_T}\gamma_M}\right) \tag{F18}$$

Next taking $\epsilon_M = \epsilon_{TM}/2$ and similarly for $\|e^{-\widetilde{B_G^2}t}\|\epsilon_T$, we have that the number of operations needed to draw a sample with the required bias is in

$$O\left(\frac{\sqrt{\|e^{-\widetilde{B_G^2}t}\|\alpha}t^{3/2}D\log(f_{\max}/\epsilon_{TM})}{\sqrt{\epsilon_{TM}}\gamma_M}\right) \tag{F19}$$

Finally, if we set the error ϵ_{TMH} to be the error also including the bias in the mean estimate of $\dim\ker\Delta_k$ from having $t = \log(1/\epsilon)/\gamma$, we have after choosing both sources of error to be equal that the systematic error can be made less than ϵ_{TMH} using a number of operations in

$$\tilde{O}\left(\frac{\sqrt{\|e^{-\widetilde{B_G^2}\log(1/\epsilon_{TMH})/\gamma}\|\alpha}D\log(f_{\max}/\epsilon_{TMH})}{\sqrt{\epsilon_{TMH}}\gamma_M\gamma_{\min}^{3/2}}\right) \subseteq \tilde{O}\left(\frac{D\sqrt{\alpha}\log(f_{\max}/\epsilon_{TMH})}{\sqrt{\epsilon_{TMH}}\gamma_M\gamma_{\min}^{3/2}}\right) \tag{F20}$$

3. Sample bounds

We finally need to consider the sampling error ϵ_S that arises from taking only a finite number of samples. Standard probabilistic arguments show that the number of samples N_S needed to achieve a given ϵ_S scales as σ^2/ϵ_S^2. The mean-squared error ϵ^2 is then

$$\epsilon^2 = \epsilon_S^2 + \epsilon_{TMH}^2 \tag{F21}$$

As before, we choose to make the two contributions to the error equal. There is a final source of complexity that needs to be considered though. Algorithm 1 begins by drawing a valid k-simplex to start at to ensure that we are within the space of interest. This means that we need to randomly draw vertices until we find a vertex that is in a k-clique. The probability of drawing such a simplex is $|\mathrm{Cl}_k(G)|/|\mathcal{H}_k|$, which is the clique density for the graph. Thus with high probability, a number of samples proportional to the reciprocal of this will be needed. Each such sample requires clique detection, which is argued in Section III B scales as $O(|E|)$ up to logarithmic terms in k. This leads us to a cost of

$$N_{\mathrm{op}} \in \tilde{O}\left(\frac{|E|\binom{n}{k}}{|\mathrm{Cl}_k(G)|} + \frac{D\sqrt{\alpha}\log(f_{\max}/\epsilon)}{\epsilon^{5/2}}\frac{\sigma^2}{\gamma_M\gamma_{\min}^{3/2}}\right) \tag{F22}$$

We now substitute $\alpha, f_{\max}$ for variables related directly to the properties of the combinatorial Laplacian. We substitute $t \geq \log(1/\epsilon)/\gamma_{\min}$ throughout and drop subdominant logarithmic terms. Firstly, from (F15) we get

$$\alpha \leq 8rD\gamma_{\max}^3 \in \widetilde{\Theta}\left(\frac{Dt^{3/2}\alpha^{1/2}\gamma_{\max}^3}{\epsilon^{1/2}}\right) \implies \sqrt{\alpha} \in \widetilde{\Theta}\left(D\frac{\gamma_{\max}^3}{\gamma_{\min}^{3/2}\epsilon^{1/2}}\right) \tag{F23}$$

This bound on $\sqrt{\alpha}$, Eq. (F16), and the fact that $\widetilde{B_G^2}$ is positive semi-definite imply

$$r \in \tilde{O}\left(\|e^{-\widetilde{B_G^2}t/2}\|\sqrt{\alpha}t^{3/2}/\sqrt{\epsilon}\right) = \tilde{O}\left(D\gamma_{\max}^3/\epsilon\gamma_{\min}^3\right) \tag{F24}$$

From (F14), the logarithm scales as

$$\log\left(\frac{f_{\max}}{\epsilon}\right) \in \widetilde{\Theta}\left(\log(d_k) + \frac{D^2\gamma_{\max}^3}{\gamma_{\min}^3\epsilon}\right) \tag{F25}$$

The prior bound on the variance in (F9) evaluates to

$$\begin{aligned}\sigma^2 &\leq \frac{2^{2rD}(1/\epsilon)^{2D\max_i|\lambda_{p_i,\Gamma_i}|/\gamma_{\min}}}{d_k} \\ &\in \frac{2^{O(D^2\kappa^3/\epsilon))}(1/\epsilon)^{2D\kappa}}{d_k},\end{aligned} \tag{F26}$$

where we have defined the following quantity, which is analogous to the condition number for the Dirac operator restricted to $\mathcal{H}_k$, $\kappa = \gamma_{\max}/\gamma_{\min}$ (and neglecting the kernel). Substituting these expressions in Eq. (F22) then implies the number of operations for the algorithm obeys

$$\begin{aligned}N_{op} &\in \widetilde{O}\left(\frac{|E|\binom{n}{k}\sigma^2}{|\mathrm{Cl}_k(G)|\epsilon^2} + \frac{D^4\sigma^2\gamma_{\max}^3}{\gamma_{\min}^3\epsilon^3\gamma_M}\left(\log(d_k)D^{-2} + \frac{\gamma_{\max}^3}{\gamma^3\epsilon}\right)\right) \\ &\in \widetilde{O}\left(\frac{\sigma^2}{\epsilon^2}\left(\frac{|E|\binom{n}{k}}{|\mathrm{Cl}_k(G)|} + \frac{D^4}{\gamma_M}\frac{\kappa^3}{\epsilon}\left(\log(d_k)D^{-2} + \frac{\kappa^3}{\epsilon}\right)\right)\right).\end{aligned} \tag{F27}$$

This implies that in the event that we assume the worst case bound on the variance the total number of operations is in

$$N_{\mathrm{op}} \in \left(\frac{2^{O(D^2\kappa^3/\epsilon))}}{d_k\epsilon^{2+2D\kappa}}\left(\frac{|E|\binom{n}{k}}{|\mathrm{Cl}_k(G)|} + \frac{D^4}{\gamma_M}\frac{\kappa^3}{\epsilon}\left(\log(d_k)D^{-2} + \frac{\kappa^3}{\epsilon}\right)\right)\right) \tag{F28}$$

This shows that under worst case scenario scaling for the variance the number of operations required to estimate the dimension of the kernel within fixed multiplicative error is polynomial if the dimension of the kernel is proportional to $\binom{n}{k+1}$ and κ, D, γ_M^{-1} and $\gamma_{\max}$ are at most polylogarithmic in n.

While the above restrictions on the situations where the classical randomized algorithm is efficient are significant, they do imply that the TDA algorithm can be efficient even in cases where $\binom{n}{k+1}$ is exponentially large provided the graph is clique-dense. This possibility is not obvious if one only compares to classical algorithms like diagonalization, which scales polynomially with the dimension. Though not much is known about the behavior of κ for TDA in general, some facts are known in certain restricted cases and there are conjectures about the general case [21]. It is known for instance that for orientable simplicial complexes of dimension $d \leq 2$, $\kappa \in O(n_k^2)$ where n_k denotes the number of k-simplices and $0 \leq k \leq 2$. It is further conjectured that $\kappa \in O(n_k^{2/d})$ in most cases. Note that this latter bound is quite favourable for high-dimensional simplices, i.e. when $d \sim n$, using the approximation that $n_k \sim 2^n$.