A streamlined quantum algorithm for topological data analysis with exponentially fewer qubits

Sam McArdle[1], András Gilyén[2], and Mario Berta[3,4]

[1]AWS Center for Quantum Computing, Pasadena, CA 91125, USA
[2]Alfréd Rényi Institute of Mathematics, Budapest, Hungary
[3]Institute for Quantum Information, RWTH Aachen University, Aachen, Germany
[4]Department of Computing, Imperial College London, London, UK
30/3/26

Topological invariants of a dataset, such as the number of holes that survive from one length scale to another (persistent Betti numbers) can be used to analyze and classify data in machine learning applications. We present an improved quantum algorithm for computing persistent Betti numbers, and provide an end-to-end complexity analysis. Our approach provides large polynomial time improvements, and an exponential space saving, over existing quantum algorithms. Subject to gap dependencies, our algorithm obtains an almost quintic speedup in the number of datapoints over previously known rigorous classical algorithms for computing the persistent Betti numbers to constant additive error – the salient task for applications. However, we also introduce a quantum-inspired classical power method with provable scaling only quadratically worse than the quantum algorithm. This gives a provable classical algorithm with scaling comparable to existing classical heuristics, subject to assumptions on the gap scaling. We discuss whether quantum algorithms can achieve an exponential speedup for tasks of practical interest, as claimed previously. We conclude that there is currently no evidence for this being the case.

1 Introduction

Topological data analysis (TDA) is a method for extracting insights from large, high-dimensional, and noisy datasets [1]. TDA extends methods from algebraic topology (using group theory to classify topological objects) to datapoints sampled from an underlying topological manifold. Topological features (such as the number of connected components, or holes of a given dimension) can be good identifiers for a dataset, as they are often robust to noise and perturbations accrued when collecting the data, unlike geometric features. Topological features may be used directly to interpret the dataset – for example, to find holes in the coverage of sensors in a network [2], identify isolated areas in voter preference data [3], or classify structure in the distribution of matter in the universe [4]. They can also be used as general identifiers to compare different datasets in situations where the topology does not necessarily have an obvious interpretation – such as identifying participant groups in functional magnetic resonance imaging brain scans [5], or detecting patterns in time series data [6] to predict financial crashes [7]. Using topological features as input to machine learning models is growing increasingly popular within physics [8] (such as the unsupervised detection of phase transitions in lattice models [9, 10]) and other areas of science [11].

Classical algorithms for TDA proceed through linear algebra. An approximation of the manifold is constructed from a set of simplices (vertices, line segments, triangles, and higher dimensional generalizations) built by connecting together points within a certain length scale of each other. We are interested in computing the persistent Betti number $\beta_k^{i,j}$, the number of k-dimensional holes that survive from scale i to scale j, for a range of dimensions and length scales. The number of k-dimensional holes at a single scale is known as the Betti number

β_k^i, and is a less informative quantity [12]. A dataset consisting of N points can contain $\binom{N}{k+1}$ k-dimensional simplices, which we represent by orthonormal basis vectors. Classical algorithms for $\beta_k^{i,j}$ (and β_k^i) scale polynomially in time and space with the size of the simplicial complex, which can make them too costly to apply to systems with many datapoints, or for high k. However, large k values are likely also less prevalent in applications due to their reduced interpretability [8].

A quantum algorithm for computing Betti numbers has been proposed [13], experimentally demonstrated as a proof-of-principle [25, 26], refined [15, 16, 27], and recently extended to the more informative persistent Betti numbers [17] – with claims of an exponential speedup over classical algorithms. The quantum algorithm exploits the ability of N qubits to represent and perform linear algebra in a vector space of dimension 2^N, in some cases with cost polynomial in N. However, the quantum algorithm actually returns the (persistent) Betti number, normalized by the number of k-simplices in the complex. While tasks closely related to normalized Betti number estimation have been shown to be DQC1-hard [27, 28] (note that the task of estimating normalized (persistent) Betti numbers of typically considered clique complexes has not yet been shown to be DQC1-hard [28]), there is currently no evidence for a regime with practical use cases. When using quantum algorithms to compute the salient quantity $\beta_k^{i,j}$, the exponential advantage is lost (a similar issue afflicts quantum algorithms for approximating the Jones polynomial [29, 30] and other #P problems [31]). Therefore, quantum algorithms for TDA should be optimised, if they are to achieve significant polynomial advantage over classical approaches for applications of interest.

2 Overview of results

We present and analyze a streamlined quantum algorithm for estimating persistent Betti numbers. We reduce the problem to estimating the normalized rank of a projector that encodes the relevant topological features of the data, and show how this problem can be efficiently solved using quantum singular value transformation (QSVT) [32]. QSVT provides efficient means for applying a polynomial function to the singular values of a matrix given by a unitary block encoding.

We show how to construct block encodings of operators that represent the topology of a simplicial complex, and how to use QSVT to transform these into projectors onto the relevant subspaces.[1] We consider two possible ways of mapping simplices to qubits; a direct approach (taken by previous quantum algorithms) that uses N qubits for N datapoints, and a compact approach, introduced herein, that uses $(k+1)\log(N)$ qubits to store a k-simplex state. The compact mapping provides an exponential space saving over classical and quantum algorithms for $k = \mathcal{O}(\text{polylog}(N))$.

For a dense simplicial complex constructed from N points in $\mathbb{R}^d$, such that the number of k-simplices scales roughly as $\binom{N}{k+1}$, we have the following main result:[2] (see Appendix D for details)

Theorem 1. *(Persistent Betti number estimation) For the above parameters, we can estimate $\beta_k^{i,j}$ to additive error Δ, with success probability greater than $(1-\eta)$ using $\mathcal{O}\left(\log\left(\frac{1}{\eta}\right)\right)$ repetitions of a quantum circuit that acts on $\mathcal{O}(\min\{k\log(N), N\})$ qubits, and has depth*

$$\tilde{\mathcal{O}}\left(\frac{N^{3/2}\sqrt{k\beta_k^{i,j}\binom{N}{k+1}}}{\Delta\Lambda_{\Pi\Pi}^{0.5}\text{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\right), \tag{1}$$

(if using the circuit acting on N qubits, then the depth is reduced by a factor of $\sqrt{k}$) where $\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}$ denote the size of the smallest non-zero singular value of the specified boundary operators, and $\Lambda_{\Pi\Pi}$ is another gap parameter defined in Sec. 5.4, which has value 1 if $i = j$.

We compare our approach to existing quantum algorithms in Table 1. Only the more recent approaches have been able to compute both regular

[1] We were motivated by Ref. [17], which also used QSVT for computing persistent Betti numbers. Our approach is similar at a high level, but we work with different and more elementary operators to encode the topology. We provide an explicit comparison between our approach and the algorithm of Ref. [17] in Sec. 7.1.

[2] As noted in Appendix D the following complexity is achieved when $k(\log(d)\log(b)+b) < N$, where b is the number of bits required to describe the coordinates.

	Algorithm	β_k^i	$\beta_k^{i,j}$	Operator encoding the topology	Gate count	Space complexity
Quantum	LGZ [13]	✓	✗[a]	Hermitian embedding of boundary operators ∂_k^i, ∂_{k+1}^i	$\mathcal{O}\left(\frac{LN^3\beta_k^i\binom{N}{k+1}}{\Delta^2 \mathrm{Min}\left(\Lambda_{\partial_k^i},\Lambda_{\partial_{k+1}^i}\right)}\right)$	$\mathcal{O}(N^2)$
	GK [15]	✓	✗	$\bigoplus_k \partial_k + \partial_k^\dagger$	$\tilde{\mathcal{O}}\left(\frac{LN^2k\sqrt{\beta_k^i\binom{N}{k+1}}}{\Delta\Lambda}\right)$	$\mathcal{O}(N)$
	UAS+ [16]	✓	✗	Combinatorial Laplacian	$\mathcal{O}\left(\frac{LN(\beta_k^i)^{1.5}\binom{N}{k+1}^{1.5}}{\Delta^3\Lambda}\right)$[b]	$\mathcal{O}(N)$
	Hay [17]	✓	✓	Persistent combinatorial Laplacian	$\tilde{\mathcal{O}}\left(\frac{LN^8k^4\beta_k^{i,j}\binom{N}{k+1}}{\Delta^2\Lambda_1^2\Lambda_2}\right)$	$\mathcal{O}(N)$
	AMS [18]	✓	✗[c]	Hermitian embedding of boundary operators	Unspecified	$\mathcal{O}(N^2)$
	BSG+ [19]	✓	✗	Hermitian embedding of boundary operators	$\tilde{\mathcal{O}}\left(\frac{LN^2\sqrt{\beta_k^i\binom{N}{k+1}}}{\Delta\Lambda}\right)$	$\mathcal{O}(N\log(N))$
	This work	✓	✓	$\partial_k^i, \partial_{k+1}^j$	$\tilde{\mathcal{O}}\left(\frac{L(Nk)^{3/2}\sqrt{\beta_k^{i,j}\binom{N}{k+1}}}{\Delta\Lambda_{\Pi\Pi}^{0.5}\mathrm{Min}\left(\Lambda_{\partial_k^i},\Lambda_{\partial_{k+1}^j}\right)}\right)$[d]	$\mathcal{O}(k\log(N))$
Classical	Textbook [20]	✓	✓	Boundary operator	$\mathcal{O}\left(\lvert S_{k+1}^j\rvert^3\right)$	$\mathcal{O}(\lvert S_{k+1}^j\rvert^2)$
	Optimised textbook [21, 22]	✓	✓	Boundary operator	$\mathcal{O}\left(\lvert S_{k+1}^j\rvert^\omega\right)$	$\mathcal{O}(\lvert S_{k+1}^j\rvert^2)$
	Heuristic sparsification [23]	✓	✓	Boundary operator	$\mathcal{O}\left(\lvert S_{k+1}^j\rvert + \lvert \bar{S}_{k+1}^j\rvert^\omega\right)$	$\mathcal{O}\left(\mathrm{Max}(\lvert S_{k+1}^j\rvert, \lvert \bar{S}_{k+1}^j\rvert^2)\right)$
	Power method [24]	✓	✗	Combinatorial Laplacian	$\tilde{\mathcal{O}}\left(\frac{L\lvert S_k^i\rvert(k^2\beta_k^i + k(\beta_k^i)^2)}{\Lambda}\right)$	$\mathcal{O}(\lvert S_k^i\rvert(k+\beta_k^i))$
	Quantum-inspired power method (This work)	✓	✓	Boundary Operator	$\mathcal{O}\left(L\beta_k^{i,j}NS\frac{\log^2\left(\frac{1}{\epsilon}\right)}{\mathrm{Min}\left(\Lambda_{\partial_k^i},\Lambda_{\partial_{k+1}^j}\right)\Lambda_{\Pi\Pi}^{0.5}} + S(\beta_k^{i,j})^2 + (\beta_k^{i,j})^3\right)$	$\mathcal{O}(S(N+\beta_k^{i,j}))$

[a] It is suggested in Ref. [14] that this approach can be used for computing persistent Betti numbers by exploiting the quantum Zeno effect. We have been unable to reproduce this line of reasoning, as we discuss in Appendix E.2.

[b] One gets this scaling via substituting $|S_k^i|$ by the upper bound $\binom{N}{k+1}$.

[c] We have been unable to verify the correctness of this approach, due to nuances associated with changing the basis from simplices in the complex at scale i to those at scale j. We discuss this in more detail in Appendix E.1.

[d] We have converted the gate depth obtained for our method to a gate count by multiplying by the spatial complexity $k\log(N)$.

Table 1: A comparison of quantum and classical algorithms for topological data analysis. We present the complexities to estimate the quantities $\beta_k^i, \beta_k^{i,j}$ to additive error Δ, at L different (pairs of) length scales. $|S_k^i|$ denotes the number of k-simplices in the complex at scale i. For classical algorithms, we do not include the cost $N\sum_{j=0}^k |S_j^i|$ of finding the k-simplices in the complex at scale i, as this is typically subleading for small k, and is usually ignored in existing classical analyses. We also do not explicitly track logarithmic factors in the classical complexities, associated with updating simplices, etc. The complexities in Refs. [13, 17, 18] were not explicitly stated in terms of the parameters that we use in this paper, hence the above bounds are our best estimates based on the subroutines used in those works. Initial quantum algorithms were only able to compute regular Betti numbers, not persistent Betti numbers. For the quantum algorithms we present the complexity for the regime which we find practically most relevant, where $k \ll N$, and $|S_k^i|$ is approximately $\binom{N}{k+1}$. The quantity Λ denotes the smallest non-zero singular value of the matrix used for encoding the topology ($\partial_k^i, \partial_{k+1}^i$, or the combinatorial Laplacian). The gap $\Lambda_{\Pi\Pi}$ appearing in our work is defined in Sec. 5.4, and equals 1 when computing the non-persistent β_k^i values. The quantity $\bar{S}_k^i$ denotes the number of k-simplices in a sparsified complex, and $\omega \in [2, 2.372)$ is the matrix multiplication exponent. We define the parameter $S := |S_{k-1}^i| + |S_k^j| + |S_{k+1}^j|$

and persistent Betti numbers – the latter being the key quantity of interest for topological data analysis [12]. Prior works took more circuitous routes to computing (persistent) Betti numbers – either by implementing the relevant projections in less efficient ways (e.g., quantum phase estima-

tion) [13, 15, 18], encoding the topology in more complex operators [13, 15, 17, 18], or using incoherent, asymptotically less efficient approaches to estimate subspace dimensions [13, 16–18, 26]. In particular, compared to the existing quantum algorithm [17] for computing $\beta_k^{i,j}$, our approach provides an exponential reduction in the number of qubits required for $k = \mathcal{O}(\text{polylog}(N))$, as well as a time complexity improvement of $\mathcal{O}\left(N^{6.5}\sqrt{\binom{N}{k+1}}\Delta^{-1}\right)$ (assuming similar overheads from the respective gaps Λ of the different operators used for encoding the topology).

For dense complexes, our quantum algorithm achieves a polynomial speedup over classical algorithms for the practically relevant task of computing the actual $\beta_k^{i,j}$ values, subject to the dependence of the gap parameters on N. The speedup could be bigger for larger values of k, but is less prominent when compared to heuristic methods that sparsify the complex. We additionally present a quantum-inspired classical algorithm based on the power method, which scales only quadratically worse in the number of simplices than our quantum algorithm, with a similar gap dependence. Hence, we expect no better than a quadratic speedup for our quantum algorithm, for the task of computing the persistent Betti number to additive error. Our quantum algorithm achieves an exponential space saving compared to classical approaches for the practically relevant case of dense complexes in the small k regime. An extended comparison between our quantum algorithm and classical approaches can be found in Sec. 7.2.

The rest of this manuscript is laid out as follows. In Sec. 3 we provide a self-contained introduction to the aspects of topological data analysis required to understand our quantum algorithm (we provide a more extended and pedagogical introduction in Appendix A). We then introduce the quantum algorithm used for computing persistent Betti numbers in Sec. 4. In Sec. 5 we outline the costs of the individual building blocks of our quantum algorithm, with detailed discussion in the Appendices. In Sec. 6 we review existing classical algorithms for computing persistent Betti numbers, and introduce our quantum-inspired classical algorithm that scales only quadratically worse than the quantum algorithm. In Sec. 7 we discuss the complexity of our quantum algorithm, and the prospects for quantum speedups for practical problems of interest. We conclude in Sec. 8.

2.1 Recent developments

Shortly after the first version of this work was released on arXiv, a number of related results were released on arXiv:

- Refs. [33–35] investigated TDA from a complexity theoretic standpoint. Ref. [34] showed that determining if the Betti number of a (clique-dense) clique complex is non-zero is NP-hard in general. Refs. [33, 35] proved a stronger result, showing that this problem is QMA_1-hard. These works rigorously show that quantum algorithms should not be expected to provide exponential speedups for (persistent) Betti number estimation.

- Ref. [19] introduced optimizations of the quantum algorithm for (non-persistent) Betti numbers, including fault-tolerant overheads. This work also presented a family of graphs with (analytically computable) large Betti numbers, that result in a superpolynomial speedup of the quantum algorithm over textbook approaches for relative error estimates.

- Ref. [36] presented a simple randomized classical algorithm for estimating normalized Betti numbers to additive error. The algorithm scales as

$$\left(\frac{N}{\lambda_{\max}}\right)^{\mathcal{O}\left(\frac{1}{\sqrt{\Lambda}}\log\left(\frac{1}{\Delta}\right)\right)} \cdot \text{poly}(N)$$

  where $\lambda_{\max}$ is the largest eigenvalue, and assuming that we can efficiently sample and check k-simplices. When $k = \Omega(N)$, the algorithm runs in polynomial time for clique complexes with constant gap Λ and error $\Delta = \Omega(1/\text{poly}(N))$ (or Δ constant and $\Lambda = \Omega(1/\log^2(N))$). These are more restrictive conditions than quantum algorithms (which can simultaneously have both $\Lambda, \Delta = \Omega(1/poly(N))$). A related randomized classical algorithm was proposed independently in Ref. [19], the complexity of which depends on the mixing time of a Markov chain that enters due to the use of importance sampling, which might be hard to bound in practical use cases.

3 Background on topological data analysis

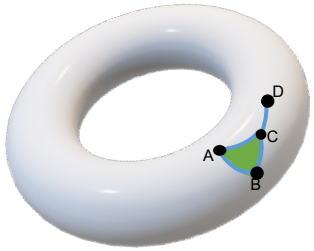


Figure 1: An example subset of datapoints arising from an underlying toroidal manifold. The datapoints (0-simplices A, B, C, D) are ordered alphabetically. After defining a length scale within which to connect datapoints, we add the relevant simplices. We show 1-simplices AB, BC, AC, CD (blue) and a 2-simplex ABC (green) that form elements of the resulting simplicial complex for this dataset. Simplices inherit the ordering of the vertices. We ascribe an orientation to the simplices based on the ordering of the vertices. An odd permutation of the vertices changes the orientation of the simplex; i.e. $AC = -CA$.

We seek to compute $\beta_k^{i,j}$ for a sufficient number of dimensions k and scales i, j so that we can capture the main topological features of the given dataset. As illustrated in Fig. 1, to understand the topology at scale i we connect datapoints within distance μ_i, generating a graph from the dataset. We use the so-called *clique complex* of the graph, where simplices correspond to the cliques in the graph: a $(k+1)$-clique defines a k-simplex. This way every $k+1$ mutually connected vertices induce a k-simplex, which we can think of as the convex hull of the corresponding data points. The set of simplices and all of their constituent faces (e.g., the 2-simplex ABC has 1-simplices AB, BC, AC and 0-simplices A, B, C as faces) at scale i is referred to as the simplicial complex S^i, while S_k^i denotes the k-simplices in S^i. Accordingly, the number of simplices in the complex at scale i is denoted by $|S^i|$, and the number of k-simplices by $|S_k^i|$.

This simplicial complex is known as a Vietoris-Rips complex, and is one possible choice [37] for approximating the underlying topological space. The maximum number of k-simplices $|S_k^i|$ is $\binom{N}{k+1}$. Mapping the simplices to orthonormal basis vectors enables the use of linear algebraic tools for understanding the topology.

The goal is to identify holes in the complex; a hole is a region of empty space, demarcated by its boundary. Studying these boundaries is crucial for finding holes, motivating the definition of a boundary operator. The k-th boundary operator maps k-simplices to their oriented boundaries. The action of the boundary operator ∂ on a k-simplex is

$$\partial[v_0, ..., v_k] = \sum_{l=0}^{k} (-1)^l [v_0, ...\hat{v}_l, ...v_k], \qquad (2)$$

where $v_0...v_k$ are the ordered vertices in the k-simplex, and $\hat{v}_l$ means the vertex is excluded from the simplex. We denote by $\partial_k^i \colon \langle S_k^i \rangle \to \langle S_{k-1}^i \rangle$ the boundary operator on the complex at length scale i restricted to the subspaces $\langle S_k^i \rangle$ and $\langle S_{k-1}^i \rangle$ spanned by the basis vectors corresponding to k and $k-1$ simplices present at scale i. For example, $\partial_1^i[CD] = D - C$, and $\partial_2^i[ABC] = BC - AC + AB$.

We can form 'chains' of k-simplices by taking linear combinations of their corresponding vectors. The boundary operator extends linearly onto chains of k-simplices. When applying the boundary operator to a closed chain of k-simplices (known as a k-cycle), each $(k-1)$-face appears twice, each time with opposite signs. For example,

$$\partial_1^i[AB + BC - AC] = B - A + C - B - C + A = 0. \qquad (3)$$

More formally, all k-cycles are in the kernel of ∂_k^i. A k-cycle can surround either empty space, or will correspond to the boundary of a $(k+1)$-chain in the complex (the example above corresponds to this latter case, as this k-cycle is given by $\partial_2^i[ABC]$). The k-cycles corresponding to the boundary of a $(k+1)$-chain in the complex are thus in the image of ∂_{k+1}^i.

To count holes in the complex at scale i, one can consider all k-cycles, and remove those that are the boundaries of $(k+1)$-chains in the complex. Accordingly, the k-th Betti number at scale i is defined [38] as

$$\beta_k^i = \dim\Big(\mathrm{Ker}(\partial_k^i)\Big) - \dim\Big(\mathrm{Im}(\partial_{k+1}^i)\Big), \qquad (4)$$

where the first term counts the number of 'hole-like' objects, and the latter removes those that are boundaries of $(k+1)$-simplices, leaving the number of true holes in the complex. Note that since ∂_ℓ^i is restricted to the subspaces $\langle S_\ell^i \rangle$ and $\langle S_{\ell-1}^i \rangle$ we have $\mathrm{Im}(\partial_{k+1}^i) \subseteq \mathrm{Ker}(\partial_k^i) \subseteq \langle S_k^i \rangle$.

Computing the Betti numbers at different scales is not sufficient to determine the persistence of topological features, as the Betti number does not uniquely identify the holes [12]. In order to determine how many holes at scale i are still present at scale j, one can consider the simplicial complexes formed at both scales (S^i, S^j), and modify Eq. (4) to give the persistent Betti numbers [20] as

$$\beta_k^{i,j} = \dim\Big(\mathrm{Ker}(\partial_k^i)\Big) - \dim\Big(\mathrm{Ker}(\partial_k^i) \cap \mathrm{Im}(\partial_{k+1}^j)\Big). \tag{5}$$

The first term in this expression again counts all hole-like objects at scale i. The second term removes all holes that were present at scale i, but have been 'filled-in' by linear combinations of $(k+1)$-simplices at scale j (which automatically includes any $(k+1)$-simplices at scale i that fill-in holes at scale i). As above $\mathrm{Ker}(\partial_k^i) \subseteq \langle S_k^i \rangle$ and $\mathrm{Im}(\partial_{k+1}^j) \subseteq \langle S_k^j \rangle$, while $\langle S_k^i \rangle \subseteq \langle S_k^j \rangle$.

Once the persistent Betti numbers have been computed for all pairs of scales, they are typically summarized in a persistence diagram, that shows when topological features are created and destroyed. Different datasets can be compared by computing the similarities between these diagrams, either using stable distance measures, or vectorization of the diagrams [1, 8, 37].

4 Quantum algorithm for persistent Betti numbers

In contrast to the approach taken herein, prior quantum algorithms did not directly use the formula in Eq. (5) for computing (persistent) Betti numbers, for a number of reasons. One such reason is that the boundary operator is not Hermitian, so one cannot directly use traditional approaches such as quantum phase estimation. Therefore, earlier works mostly computed (persistent) Betti numbers by utilizing the (persistent) combinatorial Laplacian or related operators (see Appendix A), that either made it difficult to extend methods from Betti numbers to persistent Betti numbers [13, 15, 16, 27] or resulted in higher computational complexity [17].

We reduce the problem of estimating persistent Betti numbers to that of estimating the ratio of the ranks of two orthogonal projectors. We present a quantum native approach in the QSVT framework for solving this problem. Consider the orthogonal projectors $\Pi \preceq \widetilde{\Pi}$[3]. If we treat Π as an observable and $\rho := \widetilde{\Pi}/\mathrm{rank}(\widetilde{\Pi})$ as a quantum state, then the ratio of the ranks $\frac{\mathrm{rank}(\Pi)}{\mathrm{rank}(\widetilde{\Pi})}$ equals Π's expectation value when evaluated on the mixed state ρ. We can speed up the estimation of this expectation value using amplitude estimation if we replace the mixed state ρ by its purification $|\psi\rangle$. For example, when $\widetilde{\Pi} = I$, we choose $|\psi\rangle$ to be the maximally entangled state. We require a state preparation unitary V_ψ for $|\psi\rangle$ and a block encoding [32] V_Π of Π. This leads to the following result (we provide a full version in Appendix B.1).

Theorem 2. *(Normalized projector rank estimation) Given the above ingredients, we can estimate* $\sqrt{\frac{\mathrm{rank}(\Pi)}{\mathrm{rank}(\widetilde{\Pi})}}$ *to additive error* δ *with success probability* $\geq 1-\eta$ *using* $\mathcal{O}(\log(\eta^{-1}))$ *incoherent repetitions of a quantum circuit, which makes* $\mathcal{O}\left(\frac{1}{\delta}\right)$ *calls to* V_ψ *and* V_Π *(and their inverses), and uses* $\mathcal{O}\left(\frac{a}{\delta}\right)$ *additional single- and two-qubit gates, where* a *is an upper bound on the number of qubits on which* V_ψ *and* V_Π *act.*

Proof. We employ amplitude estimation with precision δ to the following process: prepare $|\psi\rangle$ and measure Π. We repeat this process $\mathcal{O}(\log(\eta^{-1}))$ times and take the median of the estimates, which boosts the success probability exponentially due to the Chernoff bound. □

We compute (persistent) Betti numbers by the above subroutine via estimating the normalized rank of

$$\Pi_\beta := \Pi_{\mathrm{Ker}(\partial_k^i)} - \Pi_{\mathrm{Ker}(\partial_k^i) \cap \mathrm{Im}(\partial_{k+1}^j)} \preceq \Pi_{\langle S_k^i \rangle} \tag{6}$$

to sufficient precision (see Appendix B.2).

To obtain unitary block-encodings $V_{\Pi_{\mathrm{Ker}}}$ (of $\Pi_{\mathrm{Ker}(\partial_k^i)}$) and $V_{\Pi_{\mathrm{Im}}}$ (of $\Pi_{\mathrm{Im}(\partial_{k+1}^j)}$) we apply approximate threshold functions to ∂_k^i and $(\partial_{k+1}^j)^\dagger$ via QSVT, which introduces a reciprocal dependence on the smallest non-zero singular values $\Lambda_{\partial_k^i}$, $\Lambda_{\partial_{k+1}^j}$ of ∂_k^i and ∂_{k+1}^j respectively.

In general $\mathrm{Ker}(\partial_k^i) \cap \mathrm{Im}(\partial_{k+1}^j)$ is spanned by those singular vectors of $\Pi_{\mathrm{Ker}(\partial_k^i)} \cdot \Pi_{\mathrm{Im}(\partial_{k+1}^j)}$ that

[3] where $X \preceq Y$ denotes that $Y - X$ is positive semidefinite.

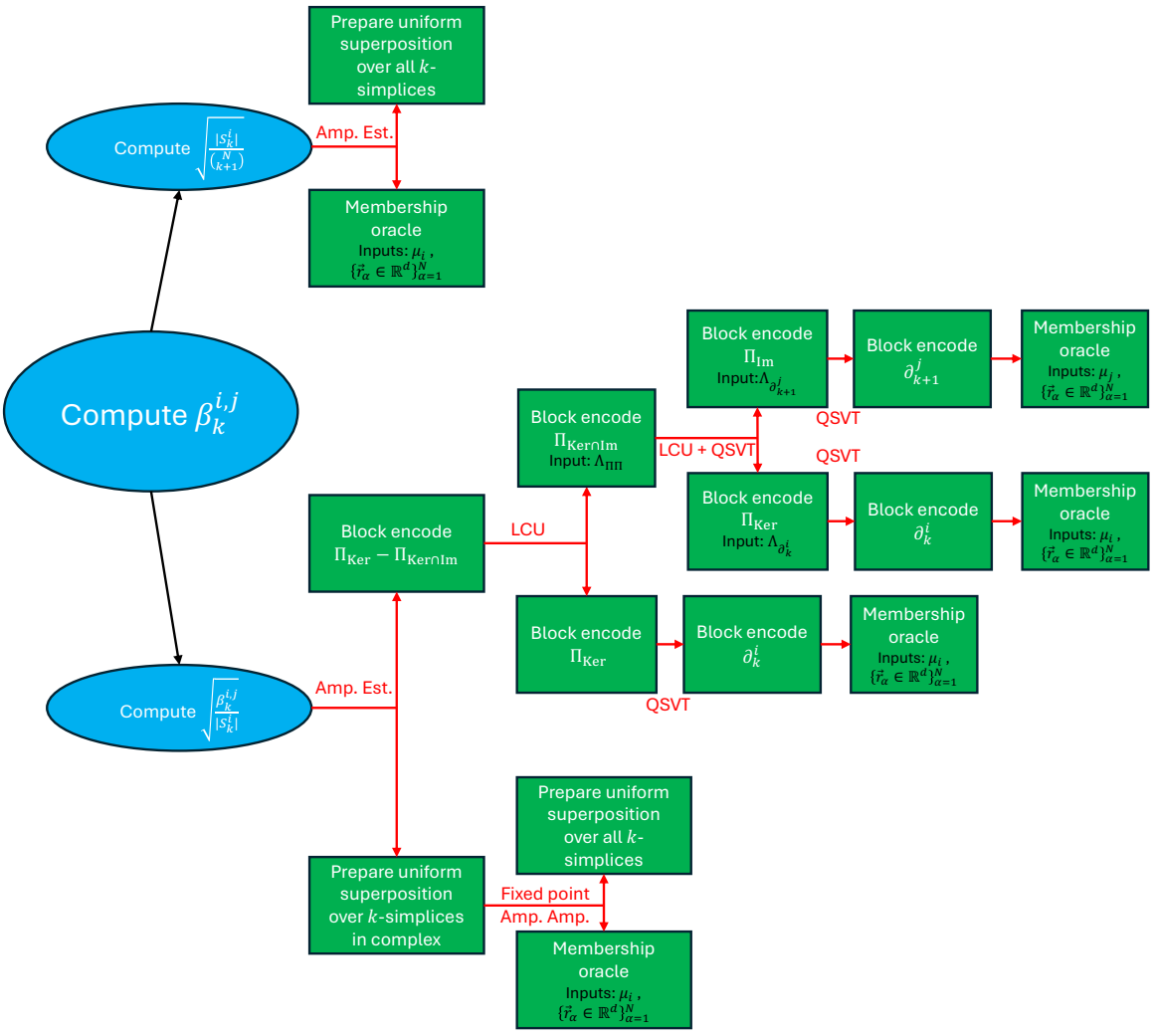


Figure 2: A flowchart illustrating the calls to made by the algorithm to various quantum and classical subroutines. Blue ovals denote classical outputs, and green boxes denote quantum subroutines. Black arrows denote classical postprocessing, while red arrows indicate coherent calls to a quantum subroutine. Some of the key quantum subroutines are named in red (Amp. Est./Amp. denotes amplitude estimation/amplification, and LCU denotes linear combination of unitaries). Quantum subroutines have their classical data inputs specified in black, with the exception of ubiquitous dependencies on the values of N, k, Δ.

have singular value 1. We can (approximately) implement a block encoding of $\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}$ by once again applying a threshold function via QSVT to $\Pi_{\mathrm{Ker}(\partial_k^i)} \cdot \Pi_{\mathrm{Im}(\partial_{k+1}^j)}$ (but now with a threshold of approximately 1). The required sharpness of the threshold is determined by the quantity $\Lambda_{\Pi\Pi}$, the deviation of the largest $\neq 1$ singular value of $\Pi_{\mathrm{Ker}(\partial_k^i)} \cdot \Pi_{\mathrm{Im}(\partial_{k+1}^j)}$ from 1. This results in an additional factor of $\Lambda_{\Pi\Pi}^{-0.5}$ (see [39, Lemma 35] for an explanation of the quadratic improvement) in the complexity of the algorithm. For ordinary Betti numbers (i.e., $i = j$) we have $\Lambda_{\Pi\Pi} = 1$ since $\mathrm{Im}(\partial_{k+1}^i) \subseteq \mathrm{Ker}(\partial_k^i)$ and thus $\Pi_{\mathrm{Ker}(\partial_k^i)} \cdot \Pi_{\mathrm{Im}(\partial_{k+1}^{j=i})} = \Pi_{\mathrm{Im}(\partial_{k+1}^i)}$.

In Fig. 2 we present a flow diagram for our quantum algorithm, showing calls to the main subroutines of the algorithm. This is complemented by the minimal pseudocode in Algorithm 1 and Algorithm 2. In order to determine the overall resource costs for estimating persistent Betti numbers using our quantum algorithm, we need to determine the cost of these subroutines. In the following section, we discuss how to implement the block encodings listed above. These require a number of ingredients, whose costs we establish for both the direct and compact mappings:

- Implement the 'membership oracle' (Algorithm 1) that determines if a given k-simplex is present in the complex at scale i, using a sequence of elementary operations, including those used for querying a (quantum) lookup-table that stores the coordinates of the data points.
- Obtain block encodings of ∂_k^i, ∂_{k+1}^j using elementary operations and the membership oracle.
- Obtain block encodings of Π_{Ker}, Π_{Im}, $\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}$, and Π_β using QSVT applied to block encodings of ∂_k^i, ∂_{k+1}^j and block encoding manipulations (linear combinations of block encodings and products of block encodings [39]).

5 Resource costs of building blocks

A fundamental quantum subroutine required by the algorithm is the membership oracle, for which we provide pseudocode in Algorithm 1.

Algorithm 1 Membership Oracle $\hat{O}_{m_k^i}$

Require: k-simplex s_k, length-scale μ_i
Ensure: $|s_k\rangle|a\rangle \mapsto |s_k\rangle|a \oplus (s_k \in S_k^i)\rangle$
1: Load point coordinates $\vec{r}_j$ for all vertices $j \in s_k$
2: Compute all $\binom{k+1}{2}$ pairwise distances
3: If all distances $\leq \mu_i$, then flip membership ancilla
4: Uncompute distances and unload point coordinates
5: **return** updated state

Below, we provide minimal pseudocode for the full quantum algorithm, in Algorithm 2.

Algorithm 2 Estimate Persistent Betti Number $\hat{\beta}_k^{i,j}$

Require: N datapoints in $\mathbb{R}^d$; feature dimension k; scales i, j with lengths μ_i, μ_j; target error Δ

Ensure: $\hat{\beta}_k^{i,j} = \beta_k^{i,j} \pm \Delta$

Assumptions: Gap parameters $\Lambda_{\partial_k^i}$, $\Lambda_{\partial_{k+1}^j}$, $\Lambda_{\Pi\Pi}$

Step 1: Estimate

$$\hat{X} = \sqrt{\frac{|S_k^i|}{\binom{N}{k+1}}} \pm \delta_1, \quad \delta_1 = \frac{\Delta}{4\beta_k^{i,j}} \sqrt{\frac{|S_k^i|}{\binom{N}{k+1}}}$$

$\hat{X} \leftarrow \textsc{AmpEst}\Big(\Pi_{S_k^i}, |\psi_{S_k}\rangle, O(\delta_1)\Big)$

// Block-encode $\Pi_{S_k^i}$ using the membership oracle $\hat{O}_{m_k^i}$

// Prepare $|\psi_{S_k}\rangle$ as a uniform superposition state over all k-simplices

Step 2: Estimate

$$\hat{Y} = \sqrt{\frac{\beta_k^{i,j}}{|S_k^i|}} \pm \delta_2, \quad \delta_2 = \frac{\Delta}{4\sqrt{|S_k^i|\,\beta_k^{i,j}}}$$

$\hat{Y} \leftarrow \textsc{AmpEst}\Big(\Pi_{\mathrm{Ker}} - \Pi_{\mathrm{Ker}\cap\mathrm{Im}}, |\psi_{S_k^i}\rangle, O(\delta_2)\Big)$

// Use QSVT + LCU to block-encode $\Pi_{\mathrm{Ker}} - \Pi_{\mathrm{Ker}\cap\mathrm{Im}}$, with $\tilde{\mathcal{O}}\left(\frac{\sqrt{Nk}}{\Lambda_{\Pi\Pi}^{0.5} \min(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j})}\right)$ calls to $\hat{O}_{m_k^i}$

// Prepare $|\psi_{S_k^i}\rangle$ using fixed-point amplitude amplification, which makes $\tilde{\mathcal{O}}\Big(\sqrt{\binom{N}{k+1}/|S_k^i|}\Big)$ calls $\hat{O}_{m_k^i}$

Step 3: Output estimate

$$\hat{\beta}_k^{i,j} = \binom{N}{k+1} \hat{X}^2 \hat{Y}^2$$

return $\hat{\beta}_k^{i,j}$

5.1 Mapping simplices to qubits

We consider two approaches to map simplices to quantum states: the direct mapping using $\mathcal{O}(N)$ qubits for a k-simplex on N datapoints, and the compact mapping, introduced herein, using $\mathcal{O}(k \log(N))$ qubits.

The direct mapping, considered in previous quantum algorithms [13], encodes k-simplices as Hamming weight-$(k+1)$ computational basis states of N qubits. Each qubit corresponds to a datapoint, and its value is $|1\rangle$ if and only if the vertex corresponding to the datapoint is incident to the given simplex.[4]

The compact mapping encodes a k-simplex built from an N-point dataset using $(k+1)\lceil \log(N+1) \rceil$ qubits. Each of the $(k+1)$ registers represents a vertex present in the simplex. The vertices are enumerated in binary, starting from 1. The all-0 state $|\bar{0}\rangle$ denotes the absence of a vertex, and is required for implementing the boundary operator. The ordering of the vertices in the state is the same as the ordering chosen for the datapoints.

k	Simplex	Direct	Compact
0	A	$\|1000000\rangle$	$\|001\rangle$
1	BD	$\|0101000\rangle$	$\|010\rangle\,\|100\rangle$
2	CEG	$\|0010101\rangle$	$\|011\rangle\,\|101\rangle\,\|111\rangle$

Table 2: Example mappings from simplices to qubits for simplices built from a vertex set $\{A, ..., G\}$.

For N datapoints, there are $\binom{N}{k+1}$ possible k-simplices. Classically storing generic linear combinations of these requires $\Omega(\binom{N}{k+1})$ memory, while a quantum register can be put into a superposition of all possible k-simplex states. The compact mapping can thus provide an exponential reduction in spatial resources over classical approaches, for dense complexes. The compact mapping is also an exponential improvement over the direct mapping when $k = \mathcal{O}(\mathrm{polylog}(N))$.

5.2 Membership oracle cost

In order to prepare $|\psi_{S_k^i}\rangle$ (a purification of $\Pi_{\langle S_k^i \rangle}/|S_k^i|$) and to restrict the boundary operators to act only on simplices present in the complex at a given length scale, the algorithm makes calls to a membership oracle $O_{m_k^i}$. The membership oracle determines if a given k-simplex is present in the complex at scale i (or not) based on the positions of the vertices $\{\vec{r}_\alpha\}$, and the length

[4] Here we mean that the vertex is part of the defining $(k+1)$-clique in the corresponding graph, rather than the datapoint being in the convex hull of the points geometrically corresponding to the simplex.

scale μ_i considered. The membership oracle acts as

$$O_{m_k^i}|s_k\rangle|a\rangle = \begin{cases} |s_k\rangle|a\oplus 1\rangle & \text{if } s_k \in S_k^i \\ |s_k\rangle|a\oplus 0\rangle & \text{if } s_k \notin S_k^i \end{cases} \quad (7)$$

We explicitly show how to implement the membership oracle for both the direct and compact mappings in Appendix C.1. It is often claimed that the quantum algorithm for TDA does not require a quantum lookup-table. While it is possible to implement the algorithm without it [16], typical formulations do in fact use quantum table lookups [13]. However, in contrast to other quantum machine learning algorithms, the quantum lookup-table used in TDA is not exponentially larger than the number of qubits used for storing the main register. We only require the ability to read from the quantum lookup-table, not to write to it.

Load	Space	Depth
Direct	N	$N\log(N)$
Compact (Slow)	$k\log(N)$ $+kdb_d^2$	$N\log(db_d)$ $+k(\log(d)\log(b_d)+b_d)$
Compact (Fast)	$Nkdb_d$ $+kdb_d^2$	$\log(N)$ $+k(\log(d)\log(b_d)+b_d)$

Table 3: A comparison of the asymptotic time and space complexity of implementing the membership oracle in the two encodings, using the optimal memory models for each. The datapoints are in $\mathbb{R}^d$ and each coordinate is represented with b bits, chosen sufficiently large to suppress under/overflow errors in the distance calculation. These complexities are derived in Appendix C.1.

Our membership oracle for direct mapped simplices follows the approach to clique finding in Ref. [40], and uses a slow-load quantum lookup-table that can be implemented with a purely classical memory by iterating a (classical) list of present edges, and applying a Toffoli gate on the corresponding vertices to increment a counter. The counter is used for verifying that all edges of the simplex are present.

Our membership oracle for compact mapped simplices also makes use of quantum lookup-tables [41]; either performing fast loads using many ancilla qubits or, as with the direct mapping, slow loads with few ancilla qubits and a classical memory. We coherently load the coordinates of the vertices defining the simplex, and calculate the distance between each pair of them. Comparing the distances against the length scale μ_i flags edges that are not present and yields a membership oracle implementation. The compact membership oracle also verifies that the vertices are stored in increasing order for consistency.

TDA is widely applied to high dimensional data. In order to reduce the cost of the membership oracle in the compact mapping, we can implement a classical pre-processing step. We can use the Johnson–Lindenstrauss lemma to embed the high-dimensional vertices into a lower dimensional space, while approximately maintaining the distances between vertices. The dimension d can be reduced to $\mathcal{O}(\log(N)/\epsilon_{JL}^2)$ to ensure the distances between all the points stay accurate to a factor of $(1\pm\epsilon_{JL})$.

5.3 Boundary operator cost

In this section we discuss the resource costs to construct block encodings of the boundary operator ∂_k^i and ∂_{k+1}^j according to Eq. (2).

Definition 1. *We say that a unitary matrix U is an $(\alpha,(a,b),\varepsilon)$-block-encoding of the matrix A if*

$$\left\| A - \alpha(\langle 0|^{\otimes a}\otimes I)U(|0\rangle^{\otimes b}\otimes I)\right\| \le \varepsilon. \quad (8)$$

We also use this notion in cases where $(\langle 0|^{\otimes a}\otimes I)U(|0\rangle^{\otimes b}\otimes I)$ acts on larger subspaces than A itself, in which in case, in Eq. (8), by A we mean its trivial embedding[5] into these larger subspaces. Setting $m=\max\{a,b\}$ we might call it an (α,m,ε)-block-encoding or m-qubit block-encoding for brevity.

We construct a unitary $V_{\partial_k^i}$ such that $(\langle 0|^{\otimes a}\otimes I)V_{\partial_k^i}(|0\rangle^{\otimes b}\otimes I) = \frac{1}{\alpha}\partial_k^i$ (and analogously for ∂_{k+1}^j).[6] The costs are summarized in Table 4, and are derived in Appendix C.2.

[5] By the trivial embedding we mean extending A with 0 matrix elements, so that the non-zero singular values and the corresponding singular-vector pairs are unchanged.

[6] Literally implementing the restriction of ∂_k^i to the subspaces $\langle S_k^i\rangle$ and $\langle S_{k+1}^i\rangle$ is technically difficult. Instead we implement $\bar{\partial}_k^i$ which is an extension of ∂_k^i onto the larger space defined by all $(k+1)\log(N)$ qubit states of the compact encoding (respectively N qubit states of the direct encoding), extending ∂_k^i trivially to the additional dimension so that $\mathrm{Im}(\partial_k^i)=\mathrm{Im}(\bar{\partial}_k^i)\subseteq\langle S_k^i\rangle$ and $\mathrm{Im}((\partial_k^i)^\dagger)=\mathrm{Im}((\bar{\partial}_k^i)^\dagger)\subseteq\langle S_{k+1}^i\rangle$, which is sufficient for our purposes.

	(α, m, ϵ)	Gate depth
Compact	$(\sqrt{(N+1)(k+1)}, \log(N+1)+\log(k+1)+1, 0)$	$\mathcal{O}(k \log\log(N+1))$ & $1 \times O_{m_k^i}$
Direct	$(\sqrt{N}, 2, 0)$	$\mathcal{O}(\log(N))$ & $1 \times O_{m_k^i}, O_{m_{k-1}^i}$

Table 4: Parameters for the block-encoding $V_{\partial_k^i}$ of ∂_k^i in the compact and direct mappings. The parameters for $V_{\partial_{k+1}^j}$ follow from replacing $k \to k+1$ and $i \to j$. The implementation of the membership oracles differ for the compact and direct encodings, as discussed in Sec. 5.2.

Our compact mapped approach proceeds by coherently swapping the vertex to be deleted into the final position of the register, and then setting it to $|\bar{0}\rangle$ to represent the absence of a vertex.

Our direct mapped approach relies on a previously observed correspondence between the boundary operator and second quantized fermionic operators: $\partial + \partial^\dagger = \sum_i a_i + a_i^\dagger$ [16, 28, 42, 43]. We considered existing approaches for implementing block-encodings of such operators [43, 44]. The chosen approach [43] for the direct mapping is a $(\sqrt{N}, 2, 0)$ block-encoding of ∂_k^i that can be implemented with $\mathcal{O}(\log(N))$ gate depth and two calls to the membership oracle. This can be compared with the approach of Ref. [17] that implemented a $(Nk, \mathcal{O}(\log(N), 0)$ block-encoding of ∂_k^i requiring $\Omega(N)$ depth and one call to the membership oracle.

5.4 Subspace projector cost

As discussed in Sec. 4 we use unitary block encodings $V_{\Pi_{\mathrm{Ker}}}$ (of $\Pi_{\mathrm{Ker}(\partial_k^i)}$) and $V_{\Pi_{\mathrm{Im}}}$ (of $\Pi_{\mathrm{Im}(\partial_{k+1}^j)}$), which are in turn used for building $V_{\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}}$ (of $\Pi_{\mathrm{Ker}\cap\mathrm{Im}}$). In this section we present an outline of the methods used for building these block encodings, and state their cost in Table 5 and Table 6. We refer to Appendix C.3 for details.

	(α, m, ϵ)	Costs $(\mathcal{O})$
C	$(1, \log((N+1)(k+1))+2, \epsilon_k)$	$\frac{\sqrt{(N+1)(k+1)}}{\Lambda_{\partial_k^i}} \log\left(\frac{1}{\epsilon_k}\right) \times (V_{\partial_k^i}, V_{\partial_k^i}^\dagger)$
D	$(1, 3, \epsilon_k)$	$\frac{\sqrt{N}}{\Lambda_{\partial_k^i}} \log\left(\frac{1}{\epsilon_k}\right) \times (V_{\partial_k^i}, V_{\partial_k^i}^\dagger)$

Table 5: A summary of the costs to implement the block encoding $V_{\Pi_{\mathrm{Ker}}}$ for the compact (C) and direct (D) mappings.

The QSVT circuit for implementing $V_{\Pi_{\mathrm{Ker}}}$ uses singular value threshold projectors [32], i.e., it (approximately) transforms ∂_k^i into the projector Π_{Ker} by mapping 0 singular values to 1 and non-zero singular values (that are at least Λ) to approximately 0. A technical detail is that we need to use an even-degree polynomial [32] ensuring that the left and right singular vectors coincide, so that we end up with an (approximate) orthonormal projector. Singular value threshold projectors are a generalization of eigenvalue threshold projectors that are used for example in Ref. [45], for applying a filter function that projects out eigenvectors above a given threshold. In Ref. [45], to prepare the ground state, the threshold is set lower than the gap Λ between the ground and first excited state. In our application we can use the polynomial approximations $P(x)$ of the threshold function used in either of [32, 45], but we need to carefully track the propagation of errors stemming from the approximation error ϵ_k.

We can implement $V_{\Pi_{\mathrm{Im}}}$ similarly by observing that $\mathrm{Im}(\partial_{k+1}^j)$ is the orthogonal complement of $\mathrm{Ker}((\partial_{k+1}^j)^\dagger)$. We can thus use the above construction to implement $V_{\Pi_{\mathrm{Im}}}$ by applying QSVT to $(\partial_{k+1}^j)^\dagger$ with the polynomial $1 - P(x)$ (as opposed to $P(x)$ above).

Given the above methods for implementing $V_{\Pi_{\mathrm{Ker}}}$ and $V_{\Pi_{\mathrm{Im}}}$, we can implement $V_{\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}}$. We cannot simply take a product of the two projectors, as in general they do not commute with each other (because of new simplices that enter the complex at scale j). However, we can implement $\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}$ by first taking the product of the block-encoded matrices Π_{Ker} and Π_{Im} [32], and then applying QSVT to the resulting block encoding of $\Pi_{\mathrm{Ker}} \cdot \Pi_{\mathrm{Im}}$. We apply a function that sends all $\neq 1$ singular values to 0. In reality, we can only send singular values below a threshold $(1 - \Lambda_{\Pi\Pi})$ to zero, and we choose $\Lambda_{\Pi\Pi}$ so that all $\neq 1$ singular values of $\Pi_{\mathrm{Ker}} \cdot \Pi_{\mathrm{Im}}$ are below $(1 - \Lambda_{\Pi\Pi})$. Such a thresholding costs $\mathcal{O}\left(\Lambda_{\Pi\Pi}^{-0.5} \log\left(\epsilon_p^{-1}\right)\right)$ (see [39, Lemma 35] for an

	(α, m, ϵ)	Costs
Compact	$\left(2, \log((N+1)^2(k+1)(k+2)) + 6, \epsilon_p + \epsilon_k + \frac{8}{\Lambda_{\Pi\Pi}^{0.5}} \log\left(\epsilon_p^{-1}\right)\sqrt{\epsilon_k + \epsilon_i}\right)$	$\mathcal{O}\left(\frac{1}{\Lambda_{\Pi\Pi}^{0.5}} \log\left(\frac{1}{\epsilon_p}\right)\right) \times$ $V_{\Pi_{\mathrm{Ker}}}, V_{\Pi_{\mathrm{Ker}}}^\dagger, V_{\Pi_{\mathrm{Im}}}, V_{\Pi_{\mathrm{Im}}}^\dagger$
Direct	$\left(2, 8, \epsilon_p + \epsilon_k + \frac{8}{\Lambda_{\Pi\Pi}^{0.5}} \log\left(\epsilon_p^{-1}\right)\sqrt{\epsilon_k + \epsilon_i}\right)$	

Table 6: A summary of the costs to implement the block encoding V_{Π_β} of $\Pi_\beta := \Pi_{\mathrm{Ker}(\partial_k^i)} - \Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}$.

explanation of the quadratic improvement) calls to $V_{\Pi_{\mathrm{Ker}}}$ and $V_{\Pi_{\mathrm{Im}}}$ to implement $V_{\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}}$ with precision ϵ_p.

5.5 State preparation cost

In this section we describe how to implement V_ψ preparing $|\psi_{S_k^i}\rangle$ from the all-0 state, where $|\psi_{S_k^i}\rangle$ is a purified maximally mixed state over k-simplices in the complex at scale i. We discuss our approach in more detail in Appendix C.4.

We use (fixed-point) amplitude amplification to construct an approximate version of $\widetilde{V}_\psi$ that prepares an ϵ_ψ-approximation of $|\psi_{S_k^i}\rangle$. For both compact and direct encodings the algorithm uses $\widetilde{\mathcal{O}}\left(\sqrt{\frac{\binom{N}{k+1}}{|S_k^i|}} \log\left(\frac{1}{\epsilon_\psi}\right)\right)$ calls to the membership oracle $O_{m_k^i}$, and to an operator U_{uni} (and its inverse) that prepares a purification of the maximally mixed state over all possible k-simplices.

In the direct mapping, k-simplices are Hamming weight $(k+1)$ computational basis states on N qubits. We can implement U_{uni} using the approaches in Ref. [46, 47] for preparing Dicke states, the most efficient of which has $\mathcal{O}(k \log(N))$ depth.

For the compact mapping, we can implement U_{uni} (to construct a uniform superposition of k-simplices, with correctly ordered vertices) through a multi-step process. We first place all vertex registers into an equal superposition. We then use exact amplitude amplification to convert this state to a superposition with no repeated vertices (but ignoring the ordering of vertices). We can use exact amplitude amplification, because we can classically exactly compute the probability $\binom{N}{k+1} \cdot (k+1)!/N^{k+1}$ that we do not get repeated vertices. Finally, we can use a reversible quantum sorting network [48], to obtain the desired equal superposition over all k-simplices, with their vertices correctly ordered.[7] In the practically relevant regime when $k \leq \sqrt{N}$ this requires

$$\mathcal{O}(\log(N) + k \log(k) \log\log(N)) \tag{9}$$

circuit depth and $\mathcal{O}(k \log(N))$ ancilla qubits.

6 Quantum-inspired classical algorithm for TDA

In this section we review existing classical algorithms for topological data analysis, and present a classical algorithm based on the power method, inspired by our quantum algorithm, for computing persistent Betti numbers. There are a number of classical algorithms for persistent Betti numbers. The most widely used approach [20] (referred to as the 'textbook', 'standard', or 'column' algorithm) proceeds by 'reducing' an $\left(|S_k^j| + |S_{k+1}^j|\right) \times \left(|S_{k-1}^j| + |S_k^j|\right)$ boundary matrix to identify pairs of simplices that create and (later) destroy a k-dimensional topological feature. The algorithm requires only a single repetition to compute the persistent Betti numbers at all length scales. Define a shorthand notation $|S_{k,k+1}^j| := |S_k^j| + |S_{k+1}^j|$. This algorithm has worst-case runtime of $\mathcal{O}(|S_{k,k+1}^j|^3)$ [49], although the runtime can approach linear in practice, due to sparsity in the complex [21], and its constant factors have been heavily optimised [50, 51]. The complexity can be improved to $\mathcal{O}(|S_{k,k+1}^j|^\omega)$ [21, 22] (where $\omega \in [2, 2.372)$ is the exponent for matrix multiplication).

Classical techniques have also been introduced that prune simplices from the complex. While it is NP-hard to find the maximally pruned complex [52], a number of heuristic approaches have been developed [23, 53–57] (and see Ref. [37] Secs. 3.1 and 5.2.6). If the number of simplices in the reduced complex is $|\bar{S}|$, the algorithm

[7] This process can be made even more efficient by performing reversible sorting only after fixed-point amplitude amplification has been performed to filter out simplices that are not in the complex at the chosen length scale.

of Ref. [23] reduces the complexity of computing persistent Betti numbers from $\mathcal{O}(|S^j_{k,k+1}|^\omega)$ to $\mathcal{O}(|S^j_{k,k+1}|+|\bar{S}|^\omega)$. When $|\bar{S}| \ll |S^j_{k,k+1}|$ the algorithm runs approximately linearly in the number of simplices in the original complex.

An alternative approach, which has parallels with the quantum algorithm for TDA, is to use the power method to find the dimension of the kernel of the combinatorial Laplacian matrix [24]. This approach was previously only applicable to computing Betti numbers. Like quantum algorithms for TDA, this approach also has a runtime that depends on the gap between the ground and first excited state of the operator used to encode the topology of the complex. This approach has a time complexity of $\tilde{\mathcal{O}}\left(\frac{|S^i_k|(k^2\beta^i_k+k(\beta^i_k)^2)}{\Lambda}\right)$ and a spatial complexity of $\mathcal{O}(|S^i_k|(k+\beta^i_k))$ [24]. This complexity provides an apples-to-apples comparison for using quantum algorithms to compute regular Betti numbers. Even in the dense complex regime of $|S^i_k| \sim \binom{N}{k+1}$, the quantum algorithm only achieves an approximately quadratic speedup in N. With the recent introduction of the persistent combinatorial Laplacian [58–60], the power method could also be used for computing persistent Betti numbers. However, the asymptotic cost of building the persistent combinatorial Laplacian is the same as that of diagonalizing it [59], so there would likely be no benefit from performing the power method with this object.

Inspired by our quantum algorithm for the computation of persistent Betti numbers, we present a classical power method for this same problem. At a high level, our algorithm constructs a polynomial approximation to the kernel projector. Consider a univariate polynomial $p(x) = a_3x^3+a_2x^2+a_1x+a_0$. We can rewrite the polynomial in the following way: $x(x(a_3x+a_2)+a_1)+a_0 = (xy+a_0)\circ(xy+a_1)\circ(a_3x+a_2)$, where the composition happens in variable y. Similarly in general we have $a_dx^d+\ldots+a_2x^2+a_1x+a_0 = (xy+a_0)\circ(xy+a_1)\circ\cdots\circ(a_dx+a_{d-1})$. This is known as Horner's method for evaluating polynomials. Now if X is a matrix, and we want to compute $p(X)v$ we can do so by only using d subsequent matrix vector products by evaluating $(X+a_0I)\circ(X+a_1I)\circ\cdots\circ(a_dX+a_{d-1}I)v$ from right to left. In our case, we have singular value transformations, so the above expression is modified to $(X+a_0I)\circ(X^\dagger+a_1I)\circ\cdots\circ(a_dX+a_{d-1}I)v$ with alternating X vs. $X^\dagger$. In order to realize expressions like $p(q(X))$, it is sufficient to replace X by $q(X)$ above, and implement each $q(X)$ analogously. Using the above construction, we can implement a kernel projector for a sparse matrix using a sequence of matrix-vector products. For an ϵ-accurate approximation of the projector $\Pi_{\mathrm{Ker}(\partial^i_k)\cap\mathrm{Im}(\partial^j_{k+1})}$ we require an $\mathcal{O}\left(\frac{\log^2\left(\frac{1}{\epsilon}\right)}{\mathrm{Min}\left(\Lambda_{\partial^i_k},\Lambda_{\partial^j_{k+1}}\right)\Lambda^{0.5}_{\text{Ш}}}\right)$ degree polynomial in the matrices $\partial^i_k, \partial^j_{k+1}$. A reasonable concern would be that the required polynomial could have exponentially large coefficients (in terms of the degree) in front of its monomials, and therefore that naive evaluation could yield large errors. It has been shown that this method of applying matrix polynomials is numerically stable, and so this is not the case [61].

The matrix ∂^i_k has column sparsity of $(k+1)$ and a row sparsity of $(N-k)$. Hence, the costs of each matrix-vector product are: ∂^i_k : $\mathcal{O}\left(S^i_{k-1}(N-k)\right)$, $(\partial^i_k)^\dagger$: $\mathcal{O}(S^i_k(k+1))$, ∂^j_{k+1} : $\mathcal{O}\left(S^j_k(N-k-1)\right)$, $\left(\partial^j_{k+1}\right)^\dagger$: $\mathcal{O}\left(S^j_{k+1}(k+2)\right)$. Note that we are storing a compressed vector with length $S := |S^i_{k-1}| + |S^j_k| + |S^j_{k+1}|$, which only stores the relevant simplices present in the complex at the given length scales. We choose an unnormalized initial random vector for the simplices in S^i_k (e.g. all entries $v_\alpha \in \pm 1$ if $\alpha \in S^i_k$, $v_\alpha = 0$ otherwise) and apply the projector $\Pi_{\mathrm{Ker}(\partial^i_k)\cap\mathrm{Im}(\partial^j_{k+1})}$ through the repeated matrix-vector multiplication described above. By picking a series of random initial states, applying the approximate projector, and performing Gram-Schmidt orthogonalization of the resulting vectors, we can compute the rank of $\Pi_{\mathrm{Ker}(\partial^i_k)\cap\mathrm{Im}(\partial^j_{k+1})}$, which is equal to the persistent Betti number $\beta^{i,j}_k$. This process uses $\beta^{i,j}_k$ random initial vectors, leading to a cost of

$$\mathcal{O}\left(\beta^{i,j}_k NS\frac{\log^2\left(\frac{1}{\epsilon}\right)}{\mathrm{Min}\left(\Lambda_{\partial^i_k},\Lambda_{\partial^j_{k+1}}\right)\Lambda^{0.5}_{\text{Ш}}}\right) \tag{10}$$

plus an additive cost of $\mathcal{O}\left(S\left(\beta^{i,j}_k\right)^2+\left(\beta^{i,j}_k\right)^3\right)$ to perform Gram-Schmidt orthogonalization and

computation of the eigenvalues in the subspace, and an additive cost of $\mathcal{O}\left(N \sum_{\ell=0}^{k+1} |S_\ell^j|\right)$ to find the $(k-1), k, (k+1)$-simplices in the complex at scales i, j[8]. We observe that for small $\beta_k^{i,j}$, and $S \sim \binom{N}{k+1}$, the complexity in $\beta_k^{i,j}, N, k$ is only quadratically worse than our quantum algorithm, while the complexity in the gap parameters is the same. The classical algorithm also returns the vectors representing the persistent holes. Moreover, this algorithm has a better asymptotic scaling in S than existing classical approaches [49, 59] (although the error and gap dependence are worse, and the constant factors may be high). For the task of estimating normalized persistent Betti numbers, we can consider a similar approach to that of Ref. [36], which is similar to performing a path integral. We would start from a single simplex (a computational basis state) and approximate the projector through the sequence of matrix vector multiplications. Averaging over a number of initial states returns an estimate of the normalized persistent Betti number. As this approach performs sparse-matrix-sparse-vector multiplication, it is possible to realize each multiplication with complexity independent of the number of simplices S. However, as the sparsity grows with each multiplication, we are limited in the number of steps we can take, which limits the accuracy of the method. We leave careful analysis of the complexity of this approach to future work.

7 Discussion

7.1 Quantum complexity

In Appendix D we determine the overall complexity of our quantum algorithm for computing persistent Betti numbers, using the costs of the building blocks introduced in the previous sections. Using the direct mapping, we show in Appendix D.1 that the algorithm has complexity given by the expression in Table 7, to estimate the persistent Betti number to additive error Δ with success probability $\geq 1-\eta$. The quantum circuit acts on $\mathcal{O}(N)$ qubits. Taking $|S_k^i| \sim \binom{N}{k+1}$, hiding logarithmic factors with the big-$\tilde{\mathcal{O}}$ notation, and accounting for parallel implementation of gates recovers Theorem 1. We can compare this scaling to the corresponding complexity of the quantum algorithm for computing persistent Betti numbers in Ref. [17], which we also show in Table 7. That work used QSVT to construct the persistent combinatorial Laplacian (see Appendix A for background), and to project onto its kernel, which encodes the persistent Betti numbers [59]. We see that our approach provides a large polynomial improvement in $|S_k^i|, N, k$ over the prior state-of-the-art.

Both of these quantum algorithms have dependencies on two different gap parameters; in our case, $\Lambda_{\Pi\Pi}, \mathrm{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)$ and in the case of Ref. [17] Λ_1, Λ_2 (where Λ_1 is the gap of a submatrix of $\partial_{k+1}^j \left(\partial_{k+1}^j\right)^\dagger$, and Λ_2 is the gap of the persistent combinatorial Laplacian). The gaps $\mathrm{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)$ and Λ_2 arise due to the need to project into the relevant kernel/image spaces, and their non-persistent analogues are also present when computing regular Betti numbers. However, the gaps $\Lambda_{\Pi\Pi}$ and Λ_1 have a more subtle origin, that appears to be specific to the problem of computing persistent Betti numbers. In Ref. [17], this gap arises from performing a change of basis necessary to build the persistent combinatorial Laplacian (see Appendix E for further discussion). In our algorithm, this gap arises from transforming the projector $\Pi_{\mathrm{Ker}(\partial_k^i)} \cdot \Pi_{\mathrm{Im}(\partial_{k+1}^j)}$ to $\Pi_{\mathrm{Ker}\cap\mathrm{Im}}$ by applying QSVT. This is necessary because $\Pi_{\mathrm{Ker}(\partial_k^i)}, \Pi_{\mathrm{Im}(\partial_{k+1}^j)}$ do not commute in general. In both cases, this additional complexity arises due to the new simplices that are present at stage j, that are not present at stage i. This makes it necessary to perform an effective change of basis, to ensure only the relevant simplices in j that can fill holes in i are considered. It remains an open question as to how these gap parameters scale as a function of N, k. A number of conjectures about the scaling of the gap of the combinatorial Laplacian were given in Ref. [24], but we are unaware of any proofs or further evidence towards these conjectures, or extensions to persistent operators. The scaling of the gap parameters will determine the viability of quantum algorithms for topological data analysis, and so it is critical to establish how these gaps scale for problems of interest.

[8] This additional cost is also present in the classical algorithms introduced above, but is typically neglected as it is subleading for small k.

Approach	Gate count
This work (Direct mapping)	$\tilde{\mathcal{O}}\left(\frac{N^2\log(N)\sqrt{\lvert S_k^i\rvert\beta_k^{i,j}}}{\Delta}\log\left(\frac{1}{\eta}\right)\times\left(\sqrt{\frac{\binom{N}{k+1}}{\lvert S_k^i\rvert}}+\frac{\sqrt{N}}{\Lambda_{\Pi\Pi}^{0.5}\mathrm{Min}\left(\Lambda_{\partial_k^i},\Lambda_{\partial_{k+1}^j}\right)}\right)\right)$
Ref. [17] (Direct mapping)	$\tilde{\mathcal{O}}\left(\frac{N^2\lvert S_k^i\rvert\beta_k^{i,j}}{\Delta^2}\times\left(\sqrt{\frac{\binom{N}{k+1}}{\lvert S_k^i\rvert}}+\frac{N^6k^4}{\Lambda_1^2\Lambda_2}\right)\right)$

Table 7: A comparison of the asymptotic complexity of our algorithm using the direct encoding, and the approach of Ref. [17]. We have converted our gate depth estimates into gate counts by pessimistically multiplying by a factor of N. The overall complexity of the approach of Ref. [17] is not explicitly stated in that work, and so we have used our best estimates based on the subroutines given. The gap parameters Λ_1, Λ_2 refer to the gaps of a submatrix of $\partial_{k+1}^j\left(\partial_{k+1}^j\right)^\dagger$, and of the persistent combinatorial Laplacian, respectively.

As shown in Theorem 1 and Appendix D.2, the corresponding expression for the compact mapping (with a slow-load lookup-table implementation of the membership oracle) is similar to that of our direct mapped algorithm. However, the quantum circuit acts on only $\mathcal{O}(k\log(N))$ qubits, providing an exponential reduction in the number of qubits required compared to the prior state-of-the-art [17], in addition to a polynomial improvement in gate complexity. This reduction in spatial complexity is significant; in practical regimes of interest, we can expect $k \approx 3$, $N \approx 10^6$, in which case (neglecting constant factors and subleading terms) the number of logical qubits is reduced from 10^6 to around 80.

7.2 Prospects for quantum advantage

We can consider the regimes in which quantum algorithms may provide a speedup over the classical approaches discussed in Sec. 6, including our new quantum-inspired approach. For k constant, both the quantum and classical algorithms run in polynomial time, preventing quantum algorithms from achieving an exponential speedup. For k scaling with N, and a clique dense complex, all known classical algorithms run in superpolynomial time. This is likely to be a fundamental result; for clique complexes determining if β_k^i is non-zero is QMA_1-hard [33, 35], and computing β_k^i is #P-hard [34].

When our task is to estimate $\beta_k^{i,j}$ to constant additive error Δ, all known quantum algorithms similarly scale superpolynomially (for k scaling with N). The limitation of quantum algorithms stems from their competing subroutines. The algorithms need to efficiently find simplices in the complex, so $|S_k^i|$ should only be polynomially smaller than $\binom{N}{k+1}$. On the other hand, the algorithms compute $\beta_k^{i,j}/|S_k^i|$ to additive error δ with an overhead of $poly(\delta^{-1})$. Converting the estimate of $\beta_k^{i,j}/|S_k^i|$ to an estimate of $\beta_k^{i,j}$ reintroduces the superpolynomial factor $|S_k^i|$, and eliminates the previously claimed [13, 16–18] exponential speedup when one needs to compute these quantities to constant additive error. A similar normalization issue arises in quantum algorithms for the approximating the Jones polynomial [29, 30] and other #P problems [31].

We can assess the potential polynomial speedup of the quantum algorithm over classical algorithms. Let us consider $k = 3$, which is sufficiently low-dimensional to be practically relevant, but is large enough so that it is already challenging to compute classically. We will assume that the gap parameters Λ are $\Omega(1/polylog(N))$, and note that $1/\Lambda \sim \mathcal{O}(poly(N))$ will reduce the polynomial speedup claimed below. Assuming a dense complex with $|S_{k+1}^j| \sim \binom{N}{k+1}$ this results in a worst-case scaling of approximately N^{10} classically [21]. Our quantum algorithm scales as $N^{3.5}$, with the additional dependencies on the precision and gap parameters noted in Table 1. Subject to these dependencies, this is an almost cubic speedup in N. For larger values of k, subject to the dependencies on the gap parameters, the quantum speedup approaches quintic: $\binom{N}{k+1}^{0.5}$ vs $\binom{N}{k+1}^{\omega}$. Nevertheless, in practice the classical algorithm often scales as $\mathcal{O}\left(|S_{k+1}^j|\right)$ [21], or this scaling can be achieved using heuristic sparsification methods [23]. Moreover, our quantum-inspired power method also scales linearly in the number of simplices, with an identical gap de-

pendence as the quantum algorithm. Hence, even if existing classical heuristics do not work, the quantum algorithm is limited to at most a quadratic speedup over classical approaches.

Based on the discussion above, the only prospect for large polynomial (or even superpolynomial) quantum speedups appears to be the case where $\beta_k^{i,j}$ is sufficiently large (compared to $|S_k^i|$) that estimating $\beta_k^{i,j}/|S_k^i|$ to relative error (equivalent to estimating $\beta_k^{i,j}$ to relative error) is a meaningful task. In Ref. [28] it was shown that estimating the normalized quasi-Betti numbers (which accounts for miscounting low-lying but non-zero singular values) of general cohomology groups is DQC1-hard. The hardness of estimating normalized (persistent) Betti numbers of a clique complex, subject to a gap assumption, which is the problem solved by existing quantum algorithms, (or alternatively, the hardness of estimating the normalized quasi-Betti numbers of a clique complex if we do not demand a gap) has not been established (see Ref. [28] Sec. 1.1). Even if DQC1-hardness can be proven for this problem, there is currently no evidence that real-world datasets possess sufficiently large (persistent) Betti numbers that such a calculation would be useful. For example, the largest known bounds on β_k^i of a Vietoris-Rips complex are $\mathcal{O}(N^k)$ [62] (though this bound is obtained inductively, and so may be loose). That work also provides a construction with $\beta_k = \Omega(N^{(k+1)/2})$. As discussed in Ref. [19, 34] this leads to an approximately quartic speedup for the quantum algorithm, compared to classical algorithms scaling as $\mathcal{O}\left(\binom{N}{k+1}\right)$. For complexes constructed from points sampled randomly from an underlying probability distribution on $\mathbb{R}^d$, the Betti numbers are much smaller; $\beta_k^i \in \mathcal{O}(N)$, on average [63, 64], which precludes large polynomial speedups, even for relative error estimates. We note that a family of graphs (not constructed from a Vietoris-Rips complex) were recently presented in Ref. [19] which have a sufficiently large Betti numbers and gap that quantum algorithms can achieve superpolynomial advantage for relative error Betti number estimation. Nevertheless, as these graphs are artificially constructed, such that their Betti numbers are known *a priori*, it is unlikely that similar speedups will be present in real-world datasets.

As discussed earlier, randomized classical algorithms were recently introduced [19, 36] that can also compute normalized (non-persistent) Betti numbers in polynomial time. These algorithms have worse gap/precision dependence than the quantum algorithm, but are efficient in the number of simplices. Hence, this further restricts the potential regime of advantage of quantum algorithms for topological data analysis.

8 Conclusion

We have developed a streamlined quantum algorithm for computing persistent Betti numbers, and analyzed its complexity. Our algorithm uses the QSVT framework to tackle the problem in a quantum native way, which enables improved complexities compared to existing quantum algorithms. We introduce a new compact mapping from simplices to qubits, that provides an exponential reduction in spatial resources for $k = \mathcal{O}(\text{polylog}(N))$. By compiling all steps of our algorithm down to primitive gates, we can provide a fair comparison to classical algorithms. We see that when the caveats of all known quantum algorithms are taken into account, they can achieve, at best, a gap-dependent polynomial speedup over classical algorithms for the salient problem of estimating low-dimensional persistent Betti numbers to constant additive error – in contrast to previous claims. The polynomial speedup of the quantum algorithm is contingent upon the as yet unknown dependence of the gap parameters on N. Moreover, we introduced a quantum-inspired classical algorithm that has better asymptotic dependence on the number of simplices than existing classical algorithms, and which limits the asymptotic speedup of our quantum algorithm to quadratic.

The compact mapping introduced in this work provides an exponential space saving over existing quantum and classical methods for the practically relevant regime of $k = \mathcal{O}(\text{polylog}(N))$. This exponential reduction in logical qubits over existing quantum algorithms is significant, particularly for the foreseeable future when logical qubit count will be a valuable resource. In fact, our algorithm achieves an exponential space saving over all classical algorithms for computing persistent Betti numbers[9].

[9] We note that recent work [36] provides a classical algorithm for estimating normalized Betti numbers with $\mathcal{O}(N)$ classical space complexity, the same as the classical mem-

Our algorithm provides an efficient and universal lens through which to investigate topological data analysis on quantum computers. Future work could determine its practicality by considering concrete resource estimates for problems of interest. Quantum advantage would be made more likely by identifying scenarios where high-dimensional persistent Betti numbers are of interest, or where computing the normalized persistent Betti numbers would be meaningful.

9 Acknowledgements

We thank Alex Dalzell and Eric Peterson for productive discussions on various aspects of this work, and Fernando Brandão for discussions and support throughout the project. A.G. acknowledges funding from the AWS Center for Quantum Computing. M.B. is supported by the EPSRC (Grant number EP/W032643/1).

References

[1] Gunnar Carlsson. Topological methods for data modelling. *Nature Reviews Physics*, 2 (12):697–708, 2020. DOI: 10.1038/s42254-020-00249-3.

[2] Vin De Silva and Robert Ghrist. Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, 7(1):339–358, 2007. DOI: 10.2140/agt.2007.7.339.

[3] Michelle Feng and Mason A Porter. Persistent homology of geospatial data: A case study with voting. *SIAM Review*, 63(1):67–99, 2021. DOI: 10.1137/19m1241519.

[4] Pratyush Pranav, Herbert Edelsbrunner, Rien Van de Weygaert, Gert Vegter, Michael Kerber, Bernard JT Jones, and Mathijs Wintraecken. The topology of the cosmic web in terms of persistent betti numbers. *Monthly Notices of the Royal Astronomical Society*, 465(4):4281–4310, 2017. DOI: 10.1093/mnras/stw2862.

[5] Bastian Rieck, Tristan Yates, Christian Bock, Karsten Borgwardt, Guy Wolf, Nicholas Turk-Browne, and Smita Krishnaswamy. Uncovering the topology of time-varying fmri data using cubical persistence. *Advances in neural information processing systems*, 33:6900–6912, 2020. DOI: 10.5555/3495724.3496303. URL https://dl.acm.org/doi/abs/10.5555/3495724.3496303.

[6] Jose A Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis. *Foundations of Computational Mathematics*, 15(3):799–838, 2015. DOI: 10.1007/s10208-014-9206-z.

[7] Marian Gidea and Yuri Katz. Topological data analysis of financial time series: Landscapes of crashes. *Physica A: Statistical Mechanics and its Applications*, 491:820–834, 2018. DOI: 10.1016/j.physa.2017.09.028.

[8] Daniel Leykam and Dimitris G. Angelakis. Topological data analysis and machine learning. *Advances in Physics: X*, 8(1):2202331, 2023. DOI: 10.1080/23746149.2023.2202331.

[9] Nicholas Sale, Jeffrey Giansiracusa, and Biagio Lucini. Quantitative analysis of phase transitions in two-dimensional xy models using persistent homology. *Phys. Rev. E*, 105:024121, Feb 2022. DOI: 10.1103/PhysRevE.105.024121. URL https://link.aps.org/doi/10.1103/PhysRevE.105.024121.

[10] Andrea Tirelli and Natanael C. Costa. Learning quantum phase transitions through topological data analysis. *Phys. Rev. B*, 104:235146, Dec 2021. DOI: 10.1103/PhysRevB.104.235146. URL https://link.aps.org/doi/10.1103/PhysRevB.104.235146.

[11] Felix Hensel, Michael Moor, and Bastian Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, 4:681108, 2021. DOI: 10.3389/frai.2021.681108.

[12] Niels Neumann and Sterre den Breeijen. Limitations of clustering using quantum persistent homology. *arXiv preprint arXiv:1911.10781*, 2019.

[13] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. Quantum algorithms for topological and geometric analysis of data. *Na-*

ory required for our quantum algorithm. However, such an algorithm has not yet been formulated for persistent Betti numbers.

ture Communications, 7(1):1–7, 2016. DOI: 10.1038/ncomms10138.

[14] Seth Lloyd. Quantum algorithms for topological and geometric analysis of data, 2021. URL https://youtu.be/G4t7Pdn9R6c?t=3853. Timecode: 1:04:13.

[15] Sam Gunn and Niels Kornerup. Review of a quantum algorithm for betti numbers. *arXiv preprint arXiv:1906.07673*, 2019.

[16] 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 preprint arXiv:2108.02811*, 2021.

[17] Ryu Hayakawa. Quantum algorithm for persistent Betti numbers and topological data analysis. *Quantum*, 6:873, December 2022. ISSN 2521-327X. DOI: 10.22331/q-2022-12-07-873. URL https://doi.org/10.22331/q-2022-12-07-873.

[18] Bernardo Ameneyro, Vasileios Maroulas, and George Siopsis. Quantum persistent homology. *Journal of Applied and Computational Topology*, 8:1929–1963, 2024. DOI: 10.1007/s41468-023-00160-7.

[19] Dominic W Berry, Yuan Su, Casper Gyurik, Robbie King, Joao Basso, Alexander Del Toro Barba, Abhishek Rajput, Nathan Wiebe, Vedran Dunjko, and Ryan Babbush. Analyzing prospects for quantum advantage in topological data analysis. *PRX Quantum*, 5(1):010319, 2024. DOI: 10.1103/prxquantum.5.010319.

[20] Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, Nov 2002. ISSN 1432-0444. DOI: 10.1007/s00454-002-2885-2. URL https://doi.org/10.1007/s00454-002-2885-2.

[21] Nikola Milosavljević, Dmitriy Morozov, and Primoz Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the twenty-seventh Annual Symposium on Computational Geometry*, pages 216–225, 2011. DOI: 10.1145/1998196.1998229.

[22] Nikola Milosavljevic, Dmitriy Morozov, and Primoz Skraba. Zigzag Persistent Homology in Matrix Multiplication Time. Research Report RR-7393, INRIA, September 2010. URL https://hal.inria.fr/inria-00520171.

[23] Konstantin Mischaikow and Vidit Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete & Computational Geometry*, 50(2):330–353, 2013. DOI: 10.1007/s00454-013-9529-6.

[24] Joel Friedman. Computing betti numbers via combinatorial laplacians. *Algorithmica*, 21(4):331–346, 1998. DOI: 10.1007/pl00009218.

[25] He-Liang Huang, Xi-Lin Wang, Peter P Rohde, Yi-Han Luo, You-Wei Zhao, Chang Liu, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Demonstration of topological data analysis on a quantum processor. *Optica*, 5(2):193–198, 2018. DOI: 10.1364/optica.5.000193.

[26] Ismail Yunus Akhalwaya, Shashanka Ubaru, Kenneth L. Clarkson, Mark S. Squillante, Vishnu Jejjala, Yang-Hui He, Kugendran Naidoo, Vasileios Kalantzis, and Lior Horesh. Towards quantum advantage on noisy quantum computers. 2022. URL https://arxiv.org/abs/2209.09371.

[27] Casper Gyurik, Chris Cade, and Vedran Dunjko. Towards quantum advantage via topological data analysis. *Quantum*, 6:855, 2022. DOI: 10.22331/q-2022-11-10-855.

[28] Chris Cade and P Marcos Crichigno. Complexity of supersymmetric systems and the cohomology problem. *Quantum*, 8:1325, 2024. DOI: 10.22331/q-2024-04-30-1325.

[29] Dorit Aharonov, Vaughan Jones, and Zeph Landau. A polynomial quantum algorithm for approximating the Jones polynomial. *Algorithmica*, 55(3):395–421, 2009. DOI: 10.1007/s00453-008-9168-0.

[30] Peter W Shor and Stephen P Jordan. Estimating Jones polynomials is a complete problem for one clean qubit. *Quantum Information and Computation*, 8(8–9):681–714, 2008. DOI: 10.26421/qic8.8-9-1.

[31] Magnus Bordewich, Michael Freedman, László Lovász, and D Welsh. Approximate counting and quantum computation. *Combinatorics, Probability and Computing*, 14(5–6):737–754, 2005. DOI: 10.1017/s0963548305007005.

[32] András Gilyén, Yuan Su, Guang Hao Low,

and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 193–204, 2019. DOI: 10.1145/3313276.3316366.

[33] Marcos Crichigno and Tamara Kohler. Clique homology is QMA_1-hard. *Nature Communications*, 15:9846, 2024. DOI: 10.1038/s41467-024-54118-z.

[34] Alexander Schmidhuber and Seth Lloyd. Complexity-theoretic limitations on quantum algorithms for topological data analysis. *PRX Quantum*, 4(4):040349, 2023. DOI: 10.1103/prxquantum.4.040349.

[35] Robbie King and Tamara Kohler. Gapped clique homology on weighted graphs is QMA_1-hard and contained in QMA. *SIAM Journal on Computing*, 55(1):S198–S231, 2026. DOI: 10.1137/24m1710243.

[36] Simon Apers, Sayantan Sen, and Dániel Szabó. A (simple) classical algorithm for estimating betti numbers. *Quantum*, 7:1202, 2023. DOI: 10.22331/q-2023-12-06-1202.

[37] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6: 1–38, 2017. DOI: 10.1140/epjds/s13688-017-0109-5.

[38] Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2005.

[39] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics [full version], 2018. arXiv: 1806.01838.

[40] 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. DOI: 10.1109/TQE.2020.3045692.

[41] Connor T. Hann, Gideon Lee, S.M. Girvin, and Liang Jiang. Resilience of quantum random access memory to generic noise. *PRX Quantum*, 2:020311, Apr 2021. DOI: 10.1103/PRXQuantum.2.020311. URL https://link.aps.org/doi/10.1103/PRXQuantum.2.020311.

[42] Ismail Yunus Akhalwaya, Yang-Hui He, Lior Horesh, Vishnu Jejjala, William Kirby, Kugendran Naidoo, and Shashanka Ubaru. Representation of the fermionic boundary operator. *Physical Review A*, 106(2):022407, 2022. DOI: 10.1103/physreva.106.022407.

[43] Iordanis Kerenidis and Anupam Prakash. Quantum machine learning with subspace states. *arXiv preprint arXiv:2202.00054*, 2022.

[44] Kianna Wan. Exponentially faster implementations of select (h) for fermionic hamiltonians. *Quantum*, 5:380, 2021. DOI: 10.22331/q-2021-01-12-380.

[45] Lin Lin and Yu Tong. Near-optimal ground state preparation. *Quantum*, 4:372, 2020. DOI: 10.22331/q-2020-12-14-372.

[46] Andreas Bärtschi and Stephan Eidenbenz. Deterministic preparation of dicke states. In *Fundamentals of Computation Theory*, pages 126–139. Springer International Publishing, 2019. DOI: 10.1007/978-3-030-25027-0_9. URL https://doi.org/10.1007%2F978-3-030-25027-0_9.

[47] Andreas Bärtschi and Stephan Eidenbenz. Short-depth circuits for dicke state preparation. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 87–96. IEEE, 2022. DOI: 10.1109/qce53715.2022.00027.

[48] András Gilyén. Quantum walk based search methods and algorithmic applications. Master's thesis, Eötvös Loránd University, 2014. URL http://web.cs.elte.hu/blobs/diplomamunkak/msc_mat/2014/gilyen_andras_pal.pdf.

[49] Dmitriy Morozov. Persistence algorithm takes cubic time in worst case. *BioGeometry News, Dept. Comput. Sci., Duke Univ*, 2, 2005.

[50] Chao Chen and Michael Kerber. Persistent homology computation with a twist. In *Proceedings 27th European workshop on computational geometry*, volume 11, pages 197–200, 2011.

[51] Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Dualities in persistent (co) homology. *Inverse Problems*, 27

(12):124003, 2011. DOI: 10.1088/0266-5611/27/12/124003.

[52] Michael Joswig and Marc E Pfetsch. Computing optimal morse matchings. *SIAM Journal on Discrete Mathematics*, 20(1):11–25, 2006. DOI: 10.1137/s0895480104445885.

[53] Marian Mrozek, Paweł Pilarczyk, and Natalia Żelazna. Homology algorithm based on acyclic subspace. *Computers & Mathematics with Applications*, 55(11):2395–2412, 2008. DOI: 10.1016/j.camwa.2007.08.044.

[54] Afra Zomorodian. The tidy set: a minimal simplicial set for computing homology of clique complexes. In *Proceedings of the twenty-sixth annual symposium on Computational geometry*, pages 257–266, 2010. DOI: 10.1145/1810959.1811004.

[55] Jonathan Ariel Barmak and Elias Gabriel Minian. Strong homotopy types, nerves and collapses. *Discrete & Computational Geometry*, 47(2):301–328, 2012. DOI: 10.1007/s00454-011-9357-5.

[56] Paweł Dłotko and Hubert Wagner. Simplification of complexes for persistent homology computations. *Homology, Homotopy and Applications*, 16(1):49–63, 2014. DOI: 10.4310/hha.2014.v16.n1.a3.

[57] Jean-Daniel Boissonnat, Siddharth Pritam, and Divyansh Pareek. Strong collapse and persistent homology. *Journal of Topology and Analysis*, 15(1):185–213, 2023. DOI: 10.1142/s1793525321500291.

[58] Rui Wang, Duc Duy Nguyen, and Guo-Wei Wei. Persistent spectral graph. *International journal for numerical methods in biomedical engineering*, 36(9):e3376, 2020. DOI: 10.1002/cnm.3376.

[59] Facundo Mémoli, Zhengchao Wan, and Yusu Wang. Persistent laplacians: Properties, algorithms and implications. *SIAM Journal on Mathematics of Data Science*, 4(2):858–884, 2022. DOI: 10.1137/21m1435471.

[60] Rui Wang, Rundong Zhao, Emily Ribando-Gros, Jiahui Chen, Yiying Tong, and Guo-Wei Wei. Hermes: Persistent spectral graph software. *Foundations of data science (Springfield, Mo.)*, 3(1):67, 2021. DOI: 10.3934/fods.2021006.

[61] Jared L. Aurentz, Anthony P. Austin, Michele Benzi, and Vassilis Kalantzis. Stable computation of generalized matrix functions via polynomial interpolation. *SIAM Journal on Matrix Analysis and Applications*, 40(1):210–234, 2019. DOI: 10.1137/18M1191786. URL https://doi.org/10.1137/18M1191786.

[62] Michael Goff. Extremal Betti numbers of Vietoris-Rips complexes. *Discrete & Computational Geometry*, 46(1):132–155, 2011. DOI: 10.1007/s00454-010-9274-z.

[63] Matthew Kahle. Random geometric complexes. *Discrete & Computational Geometry*, 45(3):553–573, jan 2011. DOI: 10.1007/s00454-010-9319-3. URL https://doi.org/10.1007%2Fs00454-010-9319-3.

[64] Omer Bobrowski and Matthew Kahle. Topology of random geometric complexes: a survey. *Journal of Applied and Computational Topology*, 1(3–4):331–364, 2018. DOI: 10.1007/s41468-017-0010-0.

[65] Gunnar Carlsson and Mikael Vejdemo-Johansson. *Topological Data Analysis with Applications*. Cambridge University Press, 2021. DOI: 10.1017/9781108975704.

[66] Chad Topaz. Chad's self-help homology tutorial for the simple(x) minded, 2016. URL https://drive.google.com/file/d/0B3Www1z6Tm8xV3ozTmN5RE94bDg/view?resourcekey=0-tE7y-zXFtV3OWSGmjUebYA. Last accessed 7 April 2022.

[67] Michelle Feng, Abigail Hickok, Yacoub Kureh, Mason Porter, and Chad Topaz. Connecting the dots: Discovering the 'shape' of data. *Front. Young Minds*, 9(551557), 2021. DOI: 10.3389/frym.2021.551557. URL https://kids.frontiersin.org/articles/10.3389/frym.2021.551557.

[68] Bastian Rieck. Topological data analysis for machine learning i: Algebraic topology, September 2020. URL https://www.youtube.com/watch?v=gVq_xXnwV-4. Last accessed 7 April 2022.

[69] Michelle Feng. Michelle feng: Topological techniques, February 2021. URL https://www.youtube.com/watch?v=M3TU4NmHDkM. Last accessed 7 April 2022.

[70] Lek-Heng Lim. Hodge laplacians on graphs. *SIAM Review*, 62(3):685–715, 2020. DOI: 10.1137/18m1223101.

[71] Timothy E Goldberg. Combinatorial lapla-

cians of simplicial complexes. *Senior Thesis, Bard College*, 2002.

[72] Guang Hao Low and Isaac L. Chuang. Hamiltonian simulation by uniform spectral amplification. arXiv: 1707.05391, 2017.

[73] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *Contemporary Mathematics Series*, pages 53–74. AMS, 2002. DOI: 10.1090/conm/305/05215. arXiv: quant-ph/0005055.

[74] Patrick Rall and Bryce Fuller. Amplitude Estimation from Quantum Signal Processing. *Quantum*, 7:937, March 2023. ISSN 2521-327X. DOI: 10.22331/q-2023-03-02-937. URL https://doi.org/10.22331/q-2023-03-02-937.

[75] 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. *Physical Review X*, 8(4):041015, 2018. DOI: 10.1103/physrevx.8.041015.

[76] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100 (16):160501, 2008. DOI: 10.1103/PhysRevLett.100.160501. arXiv: 0708.1879.

[77] Thomas G Draper, Samuel A Kutin, Eric M Rains, and Krysta M Svore. A logarithmic-depth quantum carry-lookahead adder. *Quantum Information and Computation*, 6(4–5):351–369, 2006. DOI: 10.26421/qic6.4-5-4.

[78] Shouvanik Chakrabarti, Rajiv Krishnakumar, Guglielmo Mazzola, Nikitas Stamatopoulos, Stefan Woerner, and William J. Zeng. A Threshold for Quantum Advantage in Derivative Pricing. *Quantum*, 5: 463, June 2021. ISSN 2521-327X. DOI: 10.22331/q-2021-06-01-463. URL https://doi.org/10.22331/q-2021-06-01-463.

[79] Guang Hao Low, Vadym Kliuchnikov, and Luke Schaeffer. Trading T gates for dirty qubits in state preparation and unitary synthesis. *Quantum*, 8:1375, 2024. DOI: 10.22331/q-2024-06-17-1375.

[80] Lin Lin and Yu Tong. Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems. *Quantum*, 4:361, 2020. DOI: 10.22331/q-2020-11-11-361. arXiv: 1910.14596.

[81] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2: 040203, Dec 2021. DOI: 10.1103/PRXQuantum.2.040203. URL https://link.aps.org/doi/10.1103/PRXQuantum.2.040203.

[82] András Pál Gilyén. *Quantum walk based search methods and algorithmic applications*. PhD thesis, MSc Thesis, Eötvös Loránd University, 2014.

[83] C Gidney. Quantum computing stack exchange, 2020. URL https://quantumcomputing.stackexchange.com/questions/11734/what-is-the-complexity-of-splitting-a-state-in

[84] S Pallister. Quantum computing stack exchange, 2022. URL https://quantumcomputing.stackexchange.com/questions/27864/creating-a-uniform-superposition-of-a-subset-o

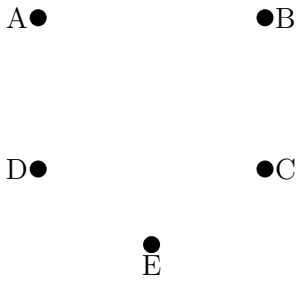
A Pedagogical introduction to topological data analysis

In this section, we provide a more pedagogical introduction to the mathematical background and notation used in topological data analysis, to build on the introduction given in the main text. This area is known as simplicial persistent homology. We will first discuss the case of computing regular Betti numbers, before moving on to persistent Betti numbers. We refer readers to Ref. [65] for a more detailed and rigorous treatment of this area. For pedagogical introductions to algebraic topology and computing persistent homology, we recommend the following review [37], notes [66], popular science article [67], and video lectures [68, 69]. We will only focus on the computation of persistent homology for point-cloud data, and do not consider extensions to graphs or pixel-based data [37].

A.1 Computing Betti numbers

Symbol	Meaning
σ_k, τ_k, s_k	Individual k-simplices
S^i	Simplicial complex at scale i defined by length scale μ_i
S^i_k	The set of k-simplices in S^i
$C_k(S^i)$	The k-th chain group of S^i_k (here, over the coefficients $\{0, \pm 1\}$)
∂^i_k	The k-th boundary operator restricted to k-simplices in the complex at scale i
H^i_k	The k-th Homology group (i.e. group of k-holes) at scale i
β^i_k	The k-th Betti number (i.e. number of k-dimensional holes) at scale i
Δ^i_k	The k-th combinatorial Laplacian at scale i

We will assume that our data consists of N points distributed in $\mathbb{R}^d$, sampled from an underlying topological manifold – for example, $N = 5$ in $\mathbb{R}^2$:



These datapoints will also be referred to as vertices, or 0-simplices. The datapoints have an associated ordering; for example, we choose to order the datapoints alphabetically by their labels. We define a length scale μ, and connect vertices by an edge if they are within μ of each other. For example, at two values of μ, we obtain:

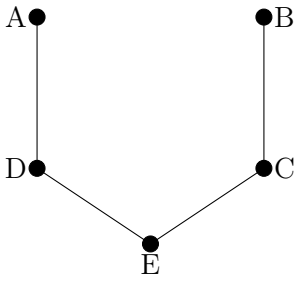
This creates a graph for each value of μ. We use the so-called *clique complex* of the graph, where simplices correspond to the cliques in the graph: a $(k+1)$-clique defines a k-simplex. This way every $(k+1)$ mutually connected vertices induce a k-simplex, which we can think of as the convex hull of the corresponding data points. For example:

Geometric realisation	Simplex	Examples
	0-Simplex	A, B
	1-Simplex	BC, CD
	2-Simplex	CDE

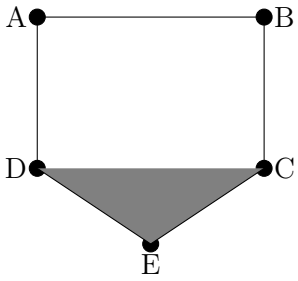
A k-simplex has cardinality $k+1$ (it contains $k+1$ vertices), and is of dimension k. For two simplices τ, σ, if $\tau \subset \sigma$, then we say τ is a face of σ. For example, the 1-simplex CD is a face of

the 2-simplex CDE. Simplices inherit the ordering of the vertices. We ascribe an orientation to the simplices based on the ordering of the vertices. An odd permutation of the vertices changes the orientation of the simplex; for example, $DA = -AD$. A set of simplices (and all of their constituent faces) is referred to as a simplicial complex S. The dimension of a simplicial complex is the highest dimension of a simplex in the complex. We define the number of simplices in the complex as $|S|$, and the number of k-simplices in the complex as $|S_k|$. As the graphs above only contain vertices and edges (0 and 1-simplices), they are 1-dimensional simplicial complexes, known as the 1-skeleton of the complex. We can then add to the complex all k-simplices represented by $(k+1)$-cliques in the graph:

$S^{\mu_1} = \{A, B, C, D, E, AD, BC, DE, CE\}$
$|S_0^{\mu_1}| = 5,\ |S_1^{\mu_1}| = 4,\ |S_2^{\mu_1}| = 0$



$S^{\mu_2} = \{A, B, C, D, E, AD, BC, DE, CE, AB, CD, CDE\}$
$|S_0^{\mu_2}| = 5,\ |S_1^{\mu_2}| = 6,\ |S_2^{\mu_2}| = 1$



This complex is known as a Vietoris-Rips complex, and provides an approximation to the underlying topological space whose homology we wish to compute. We refer the reader to Ref. [37] for discussion of other types of simplicial complexes that can be used to approximate the topological space. The maximum number of k-simplices is $\binom{N}{k+1}$. We represent the k-simplices by orthonormal basis vectors.

Under addition with field coefficients, these vectors form an abelian group. We define the k-th chain group of the complex $C_k(S)$, over field coefficients $\{-1, 0, 1\}$ (i.e. the group $\mathbb{Z}_3$ defined over addition modulo 3).

The group elements are linear combinations of k-simplices with coefficients ± 1, referred to as k-chains. For example:

$$\begin{aligned} AB + CD &\in C_1(S^{\mu_1}) \\ AB + BC + CD - AD &\in C_1(S^{\mu_2}) \end{aligned} \tag{11}$$

The simplicial complex that we have constructed acts as a scaffold that approximates the underlying topological space of interest. Promoting the simplices in the complex to vectors in a vector space enables the use of linear algebraic tools to perform the computation of simplicial homology. We are trying to identify holes in the complex; a hole is a region of empty space, demarcated by its boundary. As studying these boundaries is crucial for finding holes, we define a boundary operator.

The boundary operator ∂ is a linear map which sends k-chains to their oriented boundaries. The action of the boundary operator on a k-simplex in the complex at the given scale i is

$$\partial[v_0, ..., v_k] = \sum_{l=0}^{k} (-1)^l [v_0, ...\hat{v}_l, ...v_k] \tag{12}$$

where $v_0 ... v_k$ are the ordered vertices in the k-simplex, and $\hat{v}_l$ means the vertex is excluded from the simplex. We denote by $\partial_k^i \colon \langle S_k^i \rangle \to \langle S_{k-1}^i \rangle$ the boundary operator on the complex at length scale i restricted to the subspaces $\langle S_k^i \rangle$ and $\langle S_{k-1}^i \rangle$ spanned by the basis vectors corresponding to k and $k-1$ simplices present at scale i.

The boundary operator has the following actions on simplices in our example complex

$$\begin{aligned}\partial_1[AB] &= B - A \\ \partial_2[CDE] &= DE - CE + CD\end{aligned} \tag{13}$$

A closed cycle of k-simplices in the complex can surround either empty space, or can surround a chain of $(k+1)$-simplices in the complex. The former case corresponds to a hole, while the latter case corresponds to a hole that has been filled-in by $(k+1)$-simplices. An example of the former case in our example complex is the 1-cycle $AB+BC+CD-AD$ (where the minus sign appears due to $DA = -AD$). An example of the latter case in our example complex is the 1-cycle $DE - CE + CD = \partial_2[CDE]$. Linear combinations of these two types of k-cycles are also valid k-cycles. All k-cycles are themselves boundaryless (when applying the boundary operator to a closed cycle of k-simplices, each $(k-1)$-face appears twice, each time with opposite signs). As a result, k-cycles are objects in the kernel of ∂.

To identify the holes in the complex, we consider all possible k-cycles, and remove those that are the k-boundaries of a $(k+1)$-chain in the complex. These k-boundaries are the k-chains in the image of ∂. As both the k-cycles and k-boundaries form subgroups of $C_k(S^i)$, we can remove the filled-in holes using quotient groups (this also eliminates the issue of linear combinations of k-holes and k-boundaries being valid k-cycles - such objects are said to be homologous to the hole in question - and either the hole, or a homologous k-cycle can be chosen to represent the hole). The k-th homology group is defined as the group of k-holes in the complex

$$\mathrm{H}_k^i = \mathrm{Ker}(\partial_k^i) \;/\; \mathrm{Im}(\partial_{k+1}^i) \tag{14}$$

and the k-th Betti number is the rank of the k-th homology group

$$\begin{aligned}\beta_k^i &= \mathrm{rank}\Big(\mathrm{H}_k^i\Big) \\ &= \dim\Big(\mathrm{Ker}(\partial_k^i)\Big) - \dim\Big(\mathrm{Im}(\partial_{k+1}^i)\Big).\end{aligned} \tag{15}$$

β_0^i gives the number of connected components in S^i, β_1^i gives the number of holes, β_2^i gives the number of voids, and so on. For our example complex above, $\beta_0(S^{\mu_2}) = 1$, and $\beta_1(S^{\mu_2}) = 1$, so the complex has one connected component, and a single 1-dimensional hole.

As discussed in Ref. [70], it can be undesirable to have a number of equivalent homologous choices for the representative of the hole. The 'harmonic representative' is a chain that is orthogonal to the chains in $\mathrm{Im}(\partial_{k+1}^i)$. The group of harmonic representatives is given by $\mathrm{Ker}(\Delta_k^i)$, where

$$\Delta_k^i = \partial_{k+1}^i(\partial_{k+1}^i)^\dagger + (\partial_k^i)^\dagger\partial_k^i \tag{16}$$

is known as the k-th combinatorial Laplacian of the simplicial complex, which is a higher-order analogue of the graph Laplacian. It can be shown that [70]

$$\beta_k^i = \dim\Big(\mathrm{Ker}(\Delta_k^i)\Big). \tag{17}$$

This formulation of the problem is widely used in a number of quantum algorithms [13, 15, 16, 27]. Note that the operator $(\partial_k^i)^\dagger$ maps $C_{k-1}(S^i) \to C_k(S^i)$. The combinatorial Laplacian is Hermitian, $(k+1)$-sparse (see Ref. [71], Theorem 3.3.4), and its entries are bounded by N [27]. As discussed above, the harmonic representatives are not necessarily in the 'obvious' form. For our example complex, we find that $\mathrm{Ker}(\Delta_1(S^{\mu_2})) = -3(AB + BC + CD - AD) + \partial_2(CDE)$, rather than the obvious choice $(AB + BC + CD - AD)$. Some of the quantum algorithms [13, 15] also consider a 'k-th Dirac operator' that when squared, embeds the k-th combinatorial Laplacian in one of its sub-blocks.

The above procedures can be used for computing the Betti numbers for a fixed length scale μ_i that determines the simplicial complex at scale i. Unfortunately, it is not obvious *a priori* how the

length scale should be chosen. Moreover, computing the Betti number for different values of μ does not provide information about the persistence of topological features beyond $k = 0$. As discussed in Ref. [12], the Betti numbers do not uniquely identify topological features. For example, if there was one 1-hole created at scale μ_1, which is destroyed at scale μ_2, at the same time as a different 1-hole is created, then we would find $\beta_1(S^{\mu_1}) = \beta_1(S^{\mu_2}) = 1$, from which we would incorrectly infer that the topological feature created at μ_1 persists at length scale μ_2. In the following subsection we discuss how the methods introduced above can be extended to compute persistent Betti numbers.

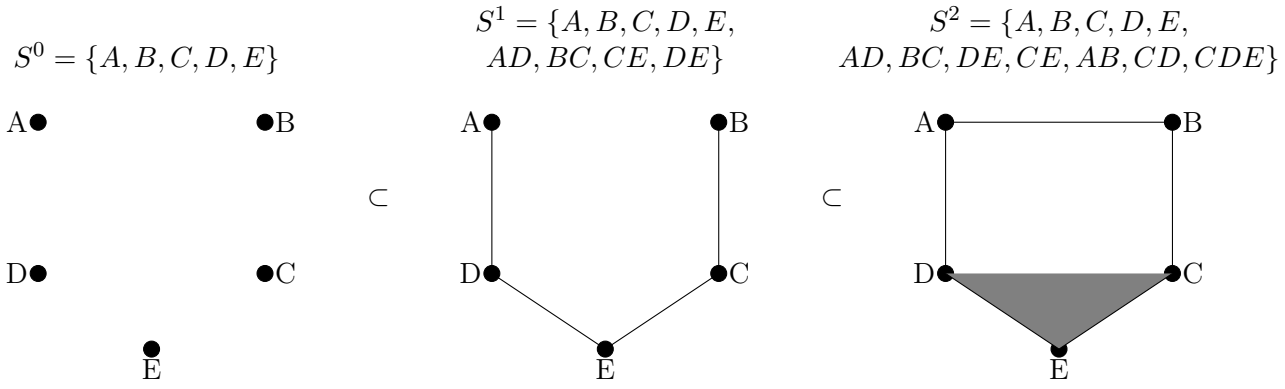
A.2 Computing persistent Betti numbers

Symbol	Meaning
$\mathrm{H}_k^{i,j}$	The $(j-i)$-th persistent k-th Homology group
$\beta_k^{i,j}$	The $(j-i)$-th persistent k-th Betti number (i.e. # k-holes present at i still present at j)
$C_{k+1}^{i,j}(S^j)$	The subgroup of $(k+1)$-chains in $C_{k+1}(S^j)$ mapped to k-chains in $C_k(S^i)$ by ∂_{k+1}^j.
$\partial_{k+1}^{i,j}$	The boundary operator ∂_{k+1} restricted to elements of $C_{k+1}^{i,j}(S^j)$.
$\Delta_k^{i,j}$	The $(j-i)$-th persistent k-th combinatorial Laplacian

In order to compute the persistent homology of the complex S, we consider a nested sequence of increasingly dense simplicial complexes – known as a filtered simplicial complex

$$S^0 \subset S^1 \subset ... \subset S^L = S \tag{18}$$

This filtration is determined by the scale μ_i; as μ_i is increased additional simplices enter the complex. If two simplices enter the complex at the same time, then we consider the lower dimensional simplex to enter first. If the two simplices are of the same dimension, an order can be assigned arbitrarily. For example, we have the following filtration:



As above, we denote the k-th boundary operator restricted to k-simplices in S^i as ∂_k^i. The k-holes in complex S^i may become 'filled in' by new $(k+1)$-simplices present in the later complex S^j. There may also be new k-holes in S^j that were not in S^i. We can count the holes in S^i that are still present in S^j by:

1. Counting all the k-holes and k-boundaries in S^i (i.e. the k-cycles $\mathrm{Ker}(\partial_k^i)$)
2. Removing those k-cycles in S^i that are k-boundaries in S^j (this is given by $\mathrm{Ker}(\partial_k^i) \cap \mathrm{Im}(\partial_{k+1}^j)$ [20])

We do not have to worry about double-counting the k-boundaries in S^i and S^j, as all k-boundaries in S^i are also k-boundaries in S^j, because all simplices in S^i are also in S^j. Formally, we can define the $(j-i)$-th persistent k-th homology group as [20]

$$\mathrm{H}_k^{i,j} := \mathrm{Ker}(\partial_k^i) \;/\; \Big(\mathrm{Ker}(\partial_k^i) \cap \mathrm{Im}(\partial_{k+1}^j)\Big) \tag{19}$$

The $(j-i)$-th persistent k-th Betti number is the rank of this group,

$$\begin{aligned}\beta_k^{i,j} &= \operatorname{rank}\Big(\mathrm{H}_k^{i,j}\Big) \\ &= \dim\Big(\operatorname{Ker}(\partial_k^i)\Big) - \dim\Big(\operatorname{Ker}(\partial_k^i) \cap \operatorname{Im}(\partial_{k+1}^j)\Big).\end{aligned} \tag{20}$$

As discussed in the main text, our quantum algorithm uses this expression for computing $\beta_k^{i,j}$, by estimating the size of these subgroups.

Recently, the task of computing persistent Betti numbers has been reformulated in terms of a persistent combinatorial Laplacian [58, 59]. Although our quantum algorithm does not proceed through this approach, the approach of Ref. [17] does, and it is instructive to compare this approach with our own. One first defines a subgroup $C_{k+1}^{i,j}(S^j)$ containing $(k+1)$-chains in the complex at scale S^j whose images under the boundary operator ∂_{k+1}^j are k-chains at scale S^i. Formally,

$$C_{k+1}^{i,j}(S^j) = \{c \in C_{k+1}(S^j) : \partial_{k+1}^j(c) \in C_k(S^i)\}. \tag{21}$$

Intuitively, the chains in $C_{k+1}^{i,j}(S^j)$ are mapped to k-cycles in $C_k(S^i)$, which can be divided into the k-boundaries already present in $C_k(S^i)$, and a subset of the k-holes in $C_k(S^i)$. In other words, the additional $(k+1)$-chains in $C_{k+1}^{i,j}(S^j)$ (beyond those already present in $C_{k+1}(S^i)$) have the action of filling-in k-holes in $C_k(S^i)$. Defining $\partial_{k+1}^{i,j}$ as ∂ restricted to the chains in $C_{k+1}^{i,j}(S^j)$ it follows that [59]

$$\operatorname{Im}(\partial_{k+1}^{i,j}) \cong \operatorname{Ker}(\partial_k^i) \cap \operatorname{Im}(\partial_{k+1}^j), \tag{22}$$

In Refs. [58, 59] the $(j-i)$-th persistent k-th combinatorial Laplacian is defined as

$$\Delta_k^{i,j} = \partial_{k+1}^{i,j} \circ (\partial_{k+1}^{i,j})^\dagger + (\partial_k^i)^\dagger \partial_k^i. \tag{23}$$

It is shown in Ref. [59] that $\beta_k^{i,j} = \dim\Big(\operatorname{Ker}(\Delta_k^{i,j})\Big)$. We have used $\circ$ in Eq. (23) to highlight that the expression of $\partial_{k+1}^{i,j}$ in terms of matrices is more complex than expressing ∂_k^i, as discussed in Ref. [59]. We elaborate more on this point in Appendix E. The reason for this complexity is that the restricted boundary operator $\partial_{k+1}^{i,j}$ must be expressed in a basis of the chains in $C_{k+1}^{i,j}(S^j)$, rather than in the basis of S_{k+1}^j. This change of basis adds complexity to both classical [59] and quantum algorithms [17]. A similar phenomenon underpins the additional round of QSVT required in our algorithm to convert $\Pi_{\mathrm{Ker}}\Pi_{\mathrm{Im}}$ to $\Pi_{\mathrm{Ker}\cap\mathrm{Im}}$, discussed in Sec. 5.4.

B Proofs of Theorems, Corollaries and Lemmas

B.1 Theorem 2: Formal version and proof

We will prove a formal version of Theorem 2, which we state here:

Theorem 3. *(Normalized projector rank estimation) Given an $(\alpha, a, \delta/4)$-block-encoding V_Π of Π and an approximate state preparation unitary $V_\psi\colon |0\rangle^{\otimes b} \mapsto |\tilde{\psi}\rangle$ such that $\big\||\tilde{\psi}\rangle - |\psi\rangle\big\| \leq \delta/4$ and $|\psi\rangle$ is a purification of a state proportional to a projector $\widetilde{\Pi} \succeq \Pi$, we can estimate the quantity $\sqrt{\frac{\operatorname{rank}(\Pi)}{\operatorname{rank}(\widetilde{\Pi})}}$ to additive error δ with success probability $\geq 1-\eta$ using $\mathcal{O}\big(\log(\eta^{-1})\big)$ incoherent repetitions of a quantum circuit, which makes $\mathcal{O}\big(\frac{\alpha}{\delta}\big)$ calls to controlled- V_ψ and V_Π (and their controlled inverses), and uses $\mathcal{O}\big((a+b)\frac{\alpha}{\delta}\big)$ additional two-qubit gates.*[10]

[10] In case $\alpha \gg 1$ and the cost of implementing V_Π is less than α times more costly than implementing V_ψ then it could be worth reducing the number of uses of V_ψ to $\mathcal{O}\big(\frac{1}{\delta}\big)$ by pre-amplifying the block-encoding V_Π by uniform eigen(singular)-value amplification [32, 72].

Proof. Let us call the purifying system B such that $\mathrm{Tr}_B(|\tilde{\psi}\rangle\langle\tilde{\psi}|) = \frac{\widetilde{\Pi}}{\mathrm{rank}(\widetilde{\Pi})}$. We consider the following process: prepare the state $|\tilde{\psi}\rangle$, apply $V_\Pi \otimes I_B$ and measure the first a qubits. The probability that we get the all-0 outcome is $\left\|(\langle 0|^{\otimes a} \otimes I)(V_\Pi \otimes I_B)(|0\rangle^{\otimes a} \otimes |\tilde{\psi}\rangle)\right\|^2$. We employ amplitude estimation with precision $\delta/(2\alpha)$ to estimate the square root of the above probability with success probability at least $2/3$. We repeat this process $\mathcal{O}(\log(\eta^{-1}))$ times and take the median of the estimates, boosting the success probability of obtaining such a precise estimate to at least $1-\eta$. This estimate, which we denote by $\hat{x}$, suffices for our purpose since

$$\sqrt{\frac{\mathrm{rank}(\Pi)}{\mathrm{rank}(\widetilde{\Pi})}} = \sqrt{\frac{\mathrm{Tr}(\Pi)}{\mathrm{rank}(\widetilde{\Pi})}} = \sqrt{\mathrm{Tr}\left(\Pi \frac{\widetilde{\Pi}}{\mathrm{rank}(\widetilde{\Pi})}\right)} = \sqrt{\langle\psi|\,(\Pi \otimes I_B)\,|\psi\rangle} = \alpha\left\|(\langle 0|^{\otimes a} \otimes I)(V_\Pi \otimes I_B)(|0\rangle^{\otimes a} \otimes |\tilde{\psi}\rangle)\right\|,$$

where the final equality only holds when V_Π is an $(\alpha, a, 0)$ block-encoding of Π and $\left\||\tilde{\psi}\rangle - |\psi\rangle\right\| = 0$. Taking into account the errors in V_Π and $|\tilde{\psi}\rangle$ yields

$$\begin{aligned}
&\sqrt{\frac{\mathrm{rank}(\Pi)}{\mathrm{rank}(\widetilde{\Pi})}} - \left\|(\Pi \otimes I_B)\,|\tilde{\psi}\rangle\right\| \\
&= \|(\Pi \otimes I_B)\,|\psi\rangle\| - \left\|(\Pi \otimes I_B)\,|\tilde{\psi}\rangle\right\| \\
&\leq \|\Pi \otimes I_B\|\left\||\psi\rangle - |\tilde{\psi}\rangle\right\| \leq \frac{\delta}{4},
\end{aligned}$$

and

$$\begin{aligned}
&\left\|(\Pi \otimes I_B)\,|\tilde{\psi}\rangle\right\| - \alpha\left\|(\langle 0|^{\otimes a} \otimes I)(V_\Pi \otimes I_B)(|0\rangle^{\otimes a} \otimes |\tilde{\psi}\rangle)\right\| \\
&\leq \left\|\Pi \otimes I_B - \alpha(\langle 0|^{\otimes a} \otimes I)(V_\Pi \otimes I_B)(|0\rangle^{\otimes a} \otimes I)\right\| \\
&= \left\|\Pi - \alpha(\langle 0|^{\otimes a} \otimes I)V_\Pi(|0\rangle^{\otimes a} \otimes I)\right\| \leq \frac{\delta}{4}.
\end{aligned}$$

Hence, including these errors and the uncertainty in $\hat{x}$ gives:

$$\begin{aligned}
&\left|\sqrt{\frac{\mathrm{rank}(\Pi)}{\mathrm{rank}(\widetilde{\Pi})}} - \alpha\hat{x}\right| \\
&\leq \left|\sqrt{\frac{\mathrm{rank}(\Pi)}{\mathrm{rank}(\widetilde{\Pi})}} - \left\|(\Pi \otimes I_B)\,|\tilde{\psi}\rangle\right\|\right| + \left|\left\|(\Pi \otimes I_B)\,|\tilde{\psi}\rangle\right\| - \alpha\hat{x}\right| \\
&\leq \frac{\delta}{4} + \left|\left\|(\Pi \otimes I_B)\,|\tilde{\psi}\rangle\right\| - \alpha\left\|(\langle 0|^{\otimes a} \otimes I)(V_\Pi \otimes I_B)(|0\rangle^{\otimes a} \otimes |\tilde{\psi}\rangle)\right\|\right| + \alpha\left|\left\|(\langle 0|^{\otimes a} \otimes I)(V_\Pi \otimes I_B)(|0\rangle^{\otimes a} \otimes |\tilde{\psi}\rangle)\right\| - \hat{x}\right| \\
&\leq \frac{\delta}{4} + \frac{\delta}{4} + \alpha\frac{\delta}{2\alpha} \\
&= \delta
\end{aligned}$$

Each elementary iteration step in amplitude estimation requires using controlled- V_Π, V_ψ and their (controlled) inverses, together with reflection operators around the initial state $|0\rangle^{\otimes(a+b)}$ and the qubits $|0\rangle^{\otimes a}$ deciding the measurement outcome. This gives the claimed gate complexity bounds on the procedure, since the cost of the final Fourier transform is dominated by that of implementing the $\mathcal{O}(\alpha/\delta)$ iteration steps. □

If amplitude estimation via textbook quantum phase estimation is used [73], we require $n = \mathcal{O}(\log(1/\delta))$ ancilla qubits, which can be a significant overhead if (exponentially) high precision is desired. Instead, one can use single ancilla variants of amplitude estimation, which can have better performance in practice [74].

B.2 Corollary 3.1: Applying Theorem 2 to estimate persistent Betti numbers

As discussed in the main text, we can estimate persistent Betti numbers using the following expression

$$\beta_k^{i,j} = \dim\Big(\mathrm{Ker}(\partial_k^i)\Big) - \dim\Big(\mathrm{Ker}(\partial_k^i) \cap \mathrm{Im}(\partial_{k+1}^j)\Big). \tag{24}$$

We can estimate this quantity, normalized by the number of k-simplices in the complex, using the quantum algorithm of Theorem 2. Hence, by rescaling the estimation error, we obtain the following corollary of (the formal version of) Theorem 2. As $\beta_k^{i,j}$ takes integer values, we can also consider the case $\beta_k^{i,j} = 0$. In this instance, we obtain a decision problem, where the goal is to distinguish between $\beta_k^{i,j} = 0$ and $\beta_k^{i,j} \neq 0$ with high certainty. In this case, we can solve the problem outlined below for $\beta_k^{i,j} = 1$, setting the precision sufficiently small (e.g., $\Delta = 1/3$) to distinguish $\beta_k^{i,j} = 0$ (running amplitude estimation and obtaining an estimate $< 2/3$) from $\beta_k^{i,j} = 1$.

Corollary 3.1. *To estimate $\beta_k^{i,j}$ to additive error Δ with success probability $\geq 1 - \eta$, we solve two instances of normalized projector rank estimation. The first encodes the value of $|S_k^i|/\binom{N}{k+1}$, and the second encodes the value of $\beta_k^{i,j}/|S_k^i|$. Each instance uses $\mathcal{O}(\log(\eta^{-1}))$ repetitions of a quantum circuit that makes $\mathcal{O}\left(\frac{\alpha_x}{\delta_x}\right)$ calls to the state preparation unitaries V_{ψ_x} (acting on b_x qubits) and the $(\alpha_x, a_x, \delta_x/4)$ block-encodings V_{Π_x} of projectors Π_x shown in Table 8. Each circuit also uses $\mathcal{O}\Big((a_x + b_x)\frac{\alpha_x}{\delta_x}\Big)$ additional two-qubit gates.*

Instance x	1	2
Estimates	$X := \sqrt{\lvert S_k^i\rvert/\binom{N}{k+1}}$	$Y := \sqrt{\beta_k^{i,j}/\lvert S_k^i\rvert}$
δ_x	$\frac{\Delta}{4\beta_k^{i,j}}\sqrt{\frac{\lvert S_k^i\rvert}{\binom{N}{k+1}}}$	$\frac{\Delta}{4\sqrt{\lvert S_k^i\rvert\beta_k^{i,j}}}$
Π_x	$\Pi_{S_k^i}$: Projects onto k-simplices in complex at scale i	$\Pi_{\mathrm{Ker}} - \Pi_{\mathrm{Ker}\cap\mathrm{Im}}$: Projects onto $\mathrm{Ker}(\partial_k^i) - \mathrm{Ker}(\partial_k^i) \cap \mathrm{Im}(\partial_{k+1}^j)$
$\lvert\psi_x\rangle$	$\lvert\psi_{S_k}\rangle$: Uniform superposition of all possible k-simplices	$\lvert\psi_{S_k^i}\rangle$: Purification of maximally mixed state over all k-simplices in complex at scale i

Table 8: Details of the error parameters and block encoded operators for each of the instances of normalized projector rank estimation used to estimate the persistent Betti number to additive error Δ using the algorithm of Theorem 2.

Proof. To prove this corollary, we define the following two random variables, which each instance of normalized projector rank estimation will estimate:

$$X := \sqrt{\frac{\mathrm{Rank}\Big(\Pi_{S_k^i}\Big)}{\binom{N}{k+1}}} \tag{25}$$

$$Y := \sqrt{\frac{\mathrm{Rank}(\Pi_{\mathrm{Ker}} - \Pi_{\mathrm{Ker}\cap\mathrm{Im}})}{\mathrm{Rank}\Big(\Pi_{S_k^i}\Big)}} \tag{26}$$

We have that $\mathrm{Rank}(\Pi_{\mathrm{Ker}} - \Pi_{\mathrm{Ker}\cap\mathrm{Im}}) = \beta_k^{i,j}$, $\mathrm{Rank}\Big(\Pi_{S_k^i}\Big) = |S_k^i|$. Hence

$$\beta_k^{i,j} = \binom{N}{k+1} X^2 Y^2. \tag{27}$$

We define the estimation errors as $\delta_\beta, \delta_x, \delta_y$, respectively. The error in $\beta_k^{i,j}$ is given by

$$\delta_\beta \leq 2\binom{N}{k+1}(\delta_x X Y^2 + \delta_y X^2 Y). \tag{28}$$

To achieve an error of $\delta_\beta = \Delta$, we set each term above equal to $\frac{\Delta}{2}$. Hence

$$\delta_x = \frac{\Delta}{4\beta_k^{i,j}} \sqrt{\frac{|S_k^i|}{\binom{N}{k+1}}} \tag{29}$$

$$\delta_y = \frac{\Delta}{4\sqrt{\beta_k^{i,j}|S_k^i|}}. \tag{30}$$

It remains to show that the specified state preparation unitaries and projector block-encodings are sufficient to encode the random variables X, Y. We focus first on X, which encodes the number of k-simplices in the complex at scale i, $|S_k^i|$. The state used is the uniform superposition over all possible k-simplices

$$\begin{aligned} |\psi_{S_k}\rangle &= \frac{1}{\sqrt{\binom{N}{k+1}}} \sum_{s_k} |s_k\rangle \\ &= \frac{1}{\sqrt{\binom{N}{k+1}}} \sum_{s_k \in S_k^i} |s_k\rangle + \frac{1}{\sqrt{\binom{N}{k+1}}} \sum_{s_k' \notin S_k^i} |s_k'\rangle . \end{aligned} \tag{31}$$

A purification is not required in this case, because we can prepare the uniform superposition in the basis in which the projector $\Pi_{S_k^i}$ is diagonal. The projector $\Pi_{S_k^i}$ projects onto k-simplices in the complex at scale i

$$\Pi_{S_k^i} = \sum_{s_k \in S_k^i} |s_k\rangle \langle s_k| . \tag{32}$$

We can verify that

$$\begin{aligned} &\left(\frac{1}{\sqrt{\binom{N}{k+1}}} \sum_{s_k \in S_k^i} \langle s_k| + \frac{1}{\sqrt{\binom{N}{k+1}}} \sum_{s_k' \notin S_k^i} \langle s_k'| \right) \left(\sum_{\sigma_k \in S_k^i} |\sigma_k\rangle \langle \sigma_k| \right) \left(\frac{1}{\sqrt{\binom{N}{k+1}}} \sum_{\tau_k \in S_k^i} |\tau_k\rangle + \frac{1}{\sqrt{\binom{N}{k+1}}} \sum_{\tau_k' \notin S_k^i} |\tau_k'\rangle \right) \\ &\qquad = \frac{1}{\binom{N}{k+1}} \sum_{s_k, \sigma_k, \tau_k \in S_k^i} \delta_{s_k,\sigma_k} \delta_{\tau_k,\sigma_k} \\ &\qquad = \frac{1}{\binom{N}{k+1}} \sum_{\sigma_k \in S_k^i} 1 \\ &\qquad = \frac{|S_k^i|}{\binom{N}{k+1}} \end{aligned}$$

as required.

For the random variable Y which encodes $\beta_k^{i,j}/|S_k^i|$, we do require the purified initial state, as this is more simple than trying to prepare a superposition over the basis in which $\Pi_{\text{Ker}} - \Pi_{\text{Ker}\cap\text{Im}}$ is diagonal. The initial state is a purified maximally mixed state over k-simplices in the complex at scale i

$$|\psi_{S_k^i}\rangle := \frac{1}{\sqrt{|S_k^i|}} \sum_{s_k \in S_k^i} |s_k\rangle |s_k\rangle . \tag{33}$$

The orthonormal basis $\{|s_k \in S_k^i\rangle\}$ is unitarily equivalent to the orthonormal basis $\{|\pi_i\rangle\}$ in which $\Pi_{\text{Ker}} - \Pi_{\text{Ker}\cap\text{Im}}$ is diagonal

$$\Pi_{\text{Ker}} - \Pi_{\text{Ker}\cap\text{Im}} := \sum_{\alpha=1}^{\beta_k^{i,j}} |\pi_\alpha\rangle \langle \pi_\alpha| . \tag{34}$$

The basis $\{|\pi_i\rangle\}$ has dimension $|S_k^i| \geq \beta_k^{i,j}$, and includes all states in the image of $\Pi_{\text{Ker}} - \Pi_{\text{Ker}\cap\text{Im}}$, and additional orthonormal basis states that are in its kernel. We thus have that

$$|\psi_{S_k^i}\rangle = \frac{1}{\sqrt{|S_k^i|}} \sum_{x=1}^{|S_k^i|} \sum_{\alpha,\beta=1}^{|S_k^i|} U_{\alpha x} |\pi_\alpha\rangle U_{\beta x} |\pi_\beta\rangle . \tag{35}$$

It can be verified that tracing out the second register yields $\widetilde{\Pi} = \frac{1}{|S_k^i|} \sum_{\alpha=1}^{|S_k^i|} |\pi_\alpha\rangle \langle\pi_\alpha|$. As a result

$$\begin{aligned}
&\langle\psi_{S_k^i}| (\Pi_{\text{Ker}} - \Pi_{\text{Ker}\cap\text{Im}}) \otimes I_B |\psi_{S_k^i}\rangle \\
&= \text{Tr}\left(\left(\frac{1}{|S_k^i|} \sum_{\alpha=1}^{|S_k^i|} |\pi_\alpha\rangle \langle\pi_\alpha| \right) \left(\sum_{\gamma=1}^{\beta_k^{i,j}} |\pi_\gamma\rangle \langle\pi_\gamma| \right) \right) \\
&= \frac{1}{|S_k^i|} \sum_{\alpha=1}^{|S_k^i|} \sum_{\gamma=1}^{\beta_k^{i,j}} \delta_{\alpha,\gamma} \\
&= \frac{\beta_k^{i,j}}{|S_k^i|}
\end{aligned}$$

as required.

The number of calls required to the specified block-encodings and state preparation unitaries follow from Theorem 2 and the above error bounds. □

C Resource costs

In this section we determine the resource costs of the individual building blocks of our algorithm.

C.1 Membership oracle construction

As discussed in the main text, the algorithm makes regular calls to a membership oracle $O_{m_k}^i$ which determines if a simplex is present in the complex at scale i (or not) based on the positions of the vertices $\{\vec{r}_\alpha\}$, and the length scale μ_i considered. The membership oracle acts as

$$O_{m_k}^i |s_k\rangle |a\rangle = |s_k\rangle |a \oplus m(s_k)\rangle \tag{36}$$

where we have defined the membership function

$$m(s_k) = \begin{cases} 1 & \text{if } s_k \in S_k^i \\ 0 & \text{if } s_k \notin S_k^i \end{cases} . \tag{37}$$

We discuss below how the membership oracle is implemented for both the compact and direct mappings.

C.1.1 Compact mapping

The membership oracle in the compact encoding checks that the vertices present in the simplex have the correct ordering, and that the distances between these vertices are less than the length scale μ_i. To implement the latter step, we will load the positions of the vertices from a quantum lookup table. The quantum lookup table requires only read-only capabilities, and can be implemented using slow loads (and few ancilla qubits) or fast loads (with many ancilla qubits). These extremes have been referred to as 'QROM' [75] and 'QRAM' [76] elsewhere in the literature, but for clarity we do not use

this terminology here, referring to the extremes as 'slow' and 'fast' instead. The quantum lookup table implements the following transformation

$$\frac{1}{\sqrt{D}}\sum_{t=0}^{D-1}|t\rangle|0\rangle^{\otimes b}\rightarrow\frac{1}{\sqrt{D}}\sum_{t=0}^{D-1}|t\rangle|X_t\rangle \tag{38}$$

where X_t are b-bit data values to be loaded. Loading D pieces of data with the slow lookup table requires $\mathcal{O}(D\log(b))$ time (with only $\mathcal{O}(D)$ non-Clifford gates [75]), and $\mathcal{O}(\log(D))$ ancilla qubits. To load D pieces of b-bit data with the fast lookup table requires $\mathcal{O}(\log(D))$ time, and $\mathcal{O}(Db)$ ancilla qubits. It is also possible to consider intermediate space-time tradeoffs [41].

The quantum lookup table is used to load the position of the vertices in the compactly encoded simplex. As a result, for a set of points in $\mathbb{R}^d$ with each coordinate stored with b_d bits, each piece of loaded data requires db_d bits. We choose b_d large enough to keep over/underflow errors introduced when calculating the squared distance between points sufficiently small. The distance between the datapoints is then calculated coherently, and used to mark any edges in the simplex that are above the length scale μ_i. If such an edge exists, the simplex does not belong in the complex. The distance checking part of the membership oracle is implemented as follows:

1. Load the coordinates of all vertices using the quantum lookup table.
2. Coherently calculate the distances between all pairs of coordinates, and verify that they are less than the length-scale μ_i.
3. Check that all distances are below threshold using multi-controlled Toffoli gates, and use to flag if the simplex is in the complex.
4. Uncompute the distances and unload the coordinates.

We first discuss how to implement steps 2 and 3. We have the following state

$$|V_0\rangle\ldots|V_k\rangle|\vec{r}_0\rangle\ldots|\vec{r}_k\rangle \tag{39}$$

on $(k+1)\log(N+1)+(k+1)db_d$ qubits. There are $k(k-1)/2\in\mathcal{O}(k^2)$ distances between pairs of vertices to be checked, which we parallelize into k rounds of k checks. We use $\mathcal{O}(kb_d)$ ancilla qubits to store the pairwise distances between k pairs of vertices, and if each distance is below the length-scale μ_i. We use a k-controlled Toffoli to flag if any of the checked distances is larger than the length-scale. We then uncompute the pairwise distances, and repeat for a different set of pairings. We require k rounds. Using the primitives shown in Table 9, we can calculate the squared distance using $\mathcal{O}(\log(d)\log(b_d)+b_d)$, using $\mathcal{O}(db_d^2)$ ancilla qubits, and compare it to the length scale using $\mathcal{O}(\log(b_d))$ depth and $\mathcal{O}(b_d)$ ancilla qubits. To do k such checks in parallel, we thus require $\mathcal{O}(\log(d)\log(b_d)+b_d)$ depth, and $\mathcal{O}(kdb_d^2)$ ancilla qubits. The overall complexity for this step is thus $\mathcal{O}(k(\log(d)\log(b_d)+b_d))$ depth, and it uses $\mathcal{O}(kdb_d^2)$ ancilla qubits (including the k-controlled Toffoli). Once all k rounds of pairwise checks have been performed, we will have k-bits that verify if all rounds of checks have passed. A k-controlled Toffoli gate on these bits flips the flag of the membership oracle.

Primitive	Depth	Ancilla qubits
Addition/Subtraction (two a-bit numbers) [77]	$\log(a)$	a
Squaring (a-bit number) [78]	a	a^2
Comparison (two a-bit numbers)	$\log(a)$	a

Table 9: Costs of primitives used for coherently calculating the squared distances between vertices.

To load the state

$$|V_0\rangle \dots |V_k\rangle \, |\vec{r}_0\rangle \dots |\vec{r}_k\rangle \tag{40}$$

on $(k+1)\log(N+1)+(k+1)db_d$ qubits, we use the quantum lookup table. Using a fast load structure, the lookup table load can be performed with $\mathcal{O}(Nkdb_d)$ gates, in $\mathcal{O}(\log(N))$ depth, and $\mathcal{O}(Nkdb_d)$ ancilla qubits. If instead we opted to use a slow load structure, there are two options available. We could either use $\mathcal{O}(k\log(N))$ ancilla qubits to perform the lookup table load of each coordinate in parallel, with a cost of $\mathcal{O}(Nkdb_d)$ gates [75] and $\mathcal{O}(N\log(db_d))$ depth [79]. A more qubit efficient approach is to loop through each vertex value ($i = 1$ to N), and on each iteration:

- XOR the value of i with each of the $(k+1)$ vertex registers.
- Using a sequence of db_d many $\log(N+1)$-controlled Toffoli gates (anti-controlled on the vertex register), write the coordinate $\vec{r}_i$ into the corresponding coordinate register. Do this in parallel for each vertex/coordinate.
- XOR the value of i with each of the $(k+1)$ vertex registers.

This approach uses no ancilla qubits (dirty qubits can be used to implement the multicontrolled Toffoli gates) and has a gate complexity of $\mathcal{O}(Nk\log(N)b_d d)$. This complexity can be further reduced to $\mathcal{O}(Nk\log\log(N+1)b_d d)$ by observing that on average between i and $i+1$ only two bits change, and so the XOR only needs to be done with these differing bits. This can also speed up the Toffoli implementation (which acts on $\log(N+1)$ qubits) by computing the OR using a tree and then using a CNOT instead of a Toffoli. The advantage is that in the OR tree updating an input bit requires only $\log\log(N+1)$ gates as opposed to the multicontrol Toffoli which requires $\mathcal{O}(\log(N))$ gates to implement. We do not explicitly use this approach in this work, as it is necessary to reformulate other subroutines to use fewer ancilla qubits, in order to make the space saving worthwhile.

To verify that the vertices have the correct ordering, we can introduce $\mathcal{O}(k)$ ancilla qubits. We use these to perform comparisons between V_i and V_{i+1} for i even. We perform all $\lceil (k+1)/2 \rceil$ comparisons in parallel. We then do the same for V_i and V_{i+1} for i odd. This requires a circuit depth of $\mathcal{O}(\log\log(N))$, and $\mathcal{O}(k\log(N))$ ancilla qubits. We can use a single $\mathcal{O}(k)$-controlled Toffoli gate to flag if the vertices have the correct ordering. This requires a circuit depth of $\mathcal{O}(k)$. As the order checking can be performed after the distance checking, we can use the same ancilla qubits for both protocols. The overall complexity for the membership oracle, factoring in both steps, is shown in Table 10.

Data loading	Depth	Ancilla qubits
Slow	$\mathcal{O}(N\log(db_d) + k(\log(d)\log(b_d) + b_d))$	$\mathcal{O}(k\log(N) + kdb_d^2)$
Fast	$\mathcal{O}(\log(N) + k(\log(d)\log(b_d) + b_d))$	$\mathcal{O}(Nkdb_d + kdb_d^2)$

Table 10: Asymptotic complexities of implementing the compact mapped membership oracle using either slow or fast quantum lookup tables to load the positions of the vertices.

C.1.2 Direct mapping

Our implementation of the membership oracle for the direct mapping follows the approach in Ref. [40] for finding cliques in graphs. We classically store a list of edges in the complex, and then iterate through this list, checking if each edge is present in the given k-simplex. This can be checked using a Toffoli gate controlled on the vertices of the edge, targeting an ancilla qubit. We then increment a counter, controlled on the value of the ancilla qubit, and then uncompute the ancilla qubit value. After all edges have been sequentially checked, we verify that the counter register contains the correct number of edges for a k-simplex ($0.5(k+1)k$), and use this to flag if the simplex is in the complex or not, and uncompute the counter register. In the worst case there are $0.5N(N-1)$ edges in the complex. By introducing $\mathcal{O}(N)$ ancilla qubits, we can check $N/2$ edges in parallel, and add their

	(α, m, ϵ)	Gate depth
Compact	$(\sqrt{(N+1)(k+1)}, \log(N+1)+\log(k+1)+1, 0)$	$\mathcal{O}(k \log\log(N+1))$ & $1 \times O_{m_k^i}$
Direct	$(\sqrt{N}, 2, 0)$	$\mathcal{O}(\log(N))$ & $1 \times O_{m_k^i}, O_{m_{k-1}^i}$

Table 11: Parameters for the block-encoding $V_{\partial_k^i}$ of ∂_k^i in the compact and direct mappings. The parameters for $V_{\partial_{k+1}^j}$ follow from replacing $k \to k+1$ and $i \to j$. The implementation of the membership oracles differ from the compact and direct encodings, as discussed in Sec. 5.2.

values (in a tree-like structure) to increment the counter. This reduces the gate depth of the circuit to $\mathcal{O}(N \log(N))$. For edge-sparse complexes, the complexity may be further reduced. A slightly more optimized implementation of the direct mapped membership oracle is presented in Ref. [19].

C.2 Boundary operator implementation

In this section we determine the resource costs to block-encode ∂_k^i and ∂_{k+1}^j. A summary of our results is presented in Table 11. Our compact mapped approach proceeds by swapping the vertex to be deleted into the final position of the register, and then setting it to $|\bar{0}\rangle$ to represent the absence of a vertex. Our direct mapped approach relies on a previously observed correspondence between the boundary operator and second quantized fermionic operators [16, 28, 42, 43], and efficient circuits for implementing such operators [42–44]. Both approaches make calls to the membership oracle, to ensure that they are restricted to simplices of the desired dimension that are present at the specified length scale.

We restate the definition of a block-encoding here for convenience:

Definition 2. *We say that a unitary matrix U is an $(\alpha, (a,b), \varepsilon)$-block-encoding of the matrix A if*

$$\left\| A - \alpha(\langle 0|^{\otimes a} \otimes I) U (|0\rangle^{\otimes b} \otimes I) \right\| \leq \varepsilon. \tag{41}$$

We also use this notion in cases where $(\langle 0|^{\otimes a} \otimes I)U(\langle 0|^{\otimes b} \otimes I)$ acts on larger subspaces than A itself, in which in Eq. (41) case by A we mean its trivial embedding[11] into these larger subspaces. Setting $c = \max\{a,b\}$ we might call it an (α, c, ε)-block-encoding or c-qubit block-encoding for brevity.

C.2.1 Compact mapping

To simplify the analysis, we choose N to be expressible as $2^m - 1$ for integer m. We reserve the state $|\bar{0}\rangle$ to express an absence of a vertex. For example, if we had vertices A, B, C, we would map these as $A = |01\rangle, B = |10\rangle, C = |11\rangle$.

We focus first on the k-th boundary operator, as the $(k+1)$-th will follow as a simple modification of the former. Under the compact mapping, we represent the k-th boundary operator at scale i as

$$\partial_k^i = \sum_{\substack{V_0 < \ldots < V_k \\ s_k \in S_k^i}} \sum_{j=0}^{k} (-1)^j \Big(|V_0\rangle \ldots |V_{j-1}\rangle |V_{j+1}\rangle \ldots |V_k\rangle \Big) \Big(\langle V_0| \ldots \langle V_j| \ldots \langle V_k| \Big) \tag{42}$$

where the ket (output) registers act on $k\log(N+1)$ qubits and the bra (input) act on $(k+1)\log(N+1)$ qubits. This operator projects from a k-simplex in the complex at scale i onto the resulting $(k-1)$-chain.

We will show how to implement an $(\alpha, (\log(N+1)+\log(k+1)+1, \log(k+1)+1), 0)$ block-encoding $V_{\partial_k^i}$ such that

$$\Big(I_{k\log(N+1)} \otimes \langle 0|^{\otimes \log(N+1)} \otimes \langle 0| \otimes \langle 0|^{\otimes \log(k+1)} \Big) V_{\partial_k^i} \Big(I_{(k+1)\log(N+1)} \otimes |0\rangle \otimes |0\rangle^{\otimes \log(k+1)} \Big) = \frac{\partial_k^i}{\alpha} \tag{43}$$

[11] By the trivial embedding we mean extending A with 0 matrix elements, so that the non-zero singular values and the corresponding singular-vector pairs are unchanged.

where α is a normalization factor determined below. Note that this ordering of the qubits means that we don't have the usual interpretation of the matrix being block-encoded in the upper-left block of V. This is not an issue, as long as we keep track of which ancilla qubits are being used for the block-encoding. We first note that we can express

$$\begin{aligned}
&I_{(k+1)\log(N+1)} \otimes |0\rangle \\
&= \sum_{V_0,\ldots V_k} |V_0\rangle \ldots |V_k\rangle \langle V_0| \ldots \langle V_k| \otimes |0\rangle \\
&= \left(\sum_{\substack{V_0<\ldots<V_k \\ s_k \in S_k^i}} |V_0\rangle \ldots |V_k\rangle \langle V_0| \ldots \langle V_k| + \sum |V_0\rangle \ldots |V_k\rangle \langle V_0| \ldots \langle V_k| \right) \otimes |0\rangle
\end{aligned}$$

where the latter sum accounts for k-simplices with incorrectly ordered vertices, or that are not present in the complex at scale i. Hence, applying the membership oracle $O_{m_k^i}$ and an X gate to this state, we obtain

$$\begin{aligned}
&XO_{m_k^i} I_{(k+1)\log(N+1)} \otimes |0\rangle \\
&= \left(\sum_{\substack{V_0<\ldots<V_k \\ s_k \in S_k^i}} |V_0\rangle \ldots |V_k\rangle \langle V_0| \ldots \langle V_k| \right) \otimes |0\rangle + A \otimes |1\rangle
\end{aligned}$$

where we have used the operator A to denote the states that do not represent simplices in the complex at scale i. The subsequent operations will not act on the ancilla qubit, and so we can use this operation to restrict the block-encoding to simplices present in the complex. As a result, to show the implementation of $V_{\partial_k^i}$, we consider the following operations applied to the (correctly ordered, present in the complex) state $|V_0\rangle \ldots |V_j\rangle \ldots |V_k\rangle \otimes |0\rangle^{\otimes \log(k+1)}$, and step through the circuit.

We first prepare a uniform superposition over the ancilla register, which can be done with $\log(k+1)$ Hadamard gates

$$\frac{1}{\sqrt{k+1}} \sum_{j=0}^{k} |V_0\rangle \ldots |V_k\rangle \otimes |j\rangle . \tag{44}$$

Using a work ancilla qubit, we perform a j-conditioned permutation that shuffles the j-th state into the final register

$$\frac{1}{\sqrt{k+1}} \sum_{j=0}^{k} |V_0\rangle \ldots |V_{j-1}\rangle |V_{j+1}\rangle \ldots |V_k\rangle |V_j\rangle \otimes |j\rangle . \tag{45}$$

This circuit requires k controlled-SWAP gates, and $2k$-comparisons of the $\log(k+1)$-bit number j. We apply a Z gate to the final bit of $|j\rangle$

$$\frac{1}{\sqrt{k+1}} \sum_{j=0}^{k} (-1)^j |V_0\rangle \ldots |V_{j-1}\rangle |V_{j+1}\rangle \ldots |V_k\rangle |V_j\rangle \otimes |j\rangle . \tag{46}$$

We then compute the value of j into another work register, by coherently checking the value of the final register against the values of each neighbouring pairs of registers (e.g. we check $|V_j\rangle$ against $|V_{l+1}\rangle, |V_l\rangle$, and so on). Because the registers store simplices in ascending order, we seek the only pair $|V_l\rangle, |V_{l+1}\rangle$ such that $V_l < V_j < V_{l+1}$. We require $\mathcal{O}(k)$ comparisons of $\log(N+1)$ qubit registers, which takes $\mathcal{O}(k \log\log(N+1))$ depth, and $\log(N+1)$ ancilla qubits. This enables us to uniquely identify

j, which lets us uncompute the initial superposition over $|j\rangle$. We then uncompute the 2nd register $|j\rangle$ using inverse of the inequality checking outlined above. This yields the state

$$\frac{1}{\sqrt{k+1}}\sum_{j=0}^{k}(-1)^j\,|V_0\rangle\ldots|V_{j-1}\rangle\,|V_{j+1}\rangle\ldots|V_k\rangle\,|V_j\rangle\otimes|0\rangle^{\otimes\log(k+1)}\,. \tag{47}$$

We then apply $\log(N+1)$ Hadamard gates to the $|V_j\rangle$ register, resulting in the state

$$\frac{1}{\sqrt{(N+1)(k+1)}}\sum_{j=0}^{k}(-1)^j\,|V_0\rangle\ldots|V_{j-1}\rangle\,|V_{j+1}\rangle\ldots|V_k\rangle\left(|\bar{0}\rangle+\sum_{x=1}^{N}\pm|x\rangle\right)\otimes|0\rangle^{\otimes\log(k+1)}\,. \tag{48}$$

We define the above sequence of operations as the circuit $V_{\partial_k^i}$. We can then consider the matrix element prescribed by the block-encoding definition

$$\begin{aligned}
&\left(I_{k\log(N+1)}\otimes\langle 0|^{\otimes\log(N+1)}\otimes\langle 0|\otimes\langle 0|^{\otimes\log(k+1)}\right)V_{\partial_k^i}\left(I_{(k+1)\log(N+1)}\otimes|0\rangle\otimes|0\rangle^{\otimes\log(k+1)}\right)\\
=&\frac{1}{\sqrt{(N+1)(k+1)}}\sum_{\substack{V_0<..V_k\\ s_k\in S_k^i}}\sum_{j=0}^{k}(-1)^j\left(I_{k\log(N+1)}\otimes\langle 0|^{\otimes\log(N+1)}\otimes\langle 0|\otimes\langle 0|^{\otimes\log(k+1)}\right)\\
&\times\left(|V_0\rangle\ldots|V_{j-1}\rangle\,|V_{j+1}\rangle\ldots|V_k\rangle\left(|\bar{0}\rangle+\sum_{x=1}^{N}\pm|x\rangle\right)\langle V_0|\ldots\langle V_k|\otimes|0\rangle\otimes|0\rangle^{\otimes\log(k+1)}+A\otimes|1\rangle\otimes|\phi\rangle\right)\\
=&\frac{1}{\sqrt{(N+1)(k+1)}}\sum_{\substack{V_0<\ldots<V_k\\ s_k\in S_k^i}}\sum_{j=0}^{k}(-1)^j\,|V_0\rangle\ldots|V_{j-1}\rangle\,|V_{j+1}\rangle\ldots|V_k\rangle\,\langle V_0|\ldots\langle V_j|\ldots\langle V_k|
\end{aligned}$$

which shows that $V_{\partial_k^i}$ is a $\left(\sqrt{(N+1)(k+1)},(\log(N+1)+\log(k+1)+1,\log(k+1)+1\right),0)$ block-encoding of ∂_k^i. Implementing this block-encoding requires a gate-depth of $\mathcal{O}(k\log\log(N+1))$ plus one call to the membership oracle $O_{m_k^i}$.

To implement a block-encoding of ∂_{k+1}^j

$$\left(I_{(k+1)\log(N+1)}\otimes\langle 0|^{\otimes\log(N+1)}\otimes\langle 0|\otimes\langle 0|^{\otimes\log(k+2)}\right)V_{\partial_{k+1}^j}\left(I_{(k+2)\log(N+1)}\otimes|0\rangle\otimes|0\rangle^{\otimes\log(k+2)}\right)=\frac{\partial_{k+1}^j}{\alpha'} \tag{49}$$

we reuse the analysis above to show that $V_{\partial_{k+1}^j}$ is a $\left(\sqrt{(N+1)(k+2)},(\log(N+1)+\log(k+2)+1,\log(k+2)+1\right)$, block-encoding of ∂_{k+1}^j, which uses a gate-depth of $\mathcal{O}(k\log\log(N+1))$ and one call to the membership oracle $O_{m_{k+1}^j}$.

C.2.2 Direct mapping

To implement the block-encoding of the boundary operators in the direct mapping, we exploit a previously observed link to second quantized fermionic creation and annihilation operators [16, 28, 42, 43]. Specifically, the unrestricted boundary operator ∂ (which acts on all possible simplices, of all dimensions) can be expressed as

$$\partial+\partial^\dagger=\sum_{i=0}^{N-1}a_i+a_i^\dagger \tag{50}$$

where $a_i^\dagger, a_i$ are second quantized fermionic creation and annihilation operators, respectively. We can take two routes to implementing a block-encoding of this operator. The first, as outlined in Ref. [43], uses a unitary circuit with no additional ancilla qubits to implement a scaled version of this operator.

This circuit has a depth of $\mathcal{O}(\log(N))$ (composed of $2(N-1)$ single-qubit rotations), and introduces a scaling factor of $\alpha = \sqrt{N}$. Alternatively, one could use techniques for implementing block-encodings of fermionic operators introduced in the literature on quantum chemistry. For example, the approach of Ref. [44] would use $\log(N)$ ancilla qubits, and a T gate depth of $24\lceil\log(N)\rceil$ (with a T count of $24(N-1)$) to implement a block-encoding of $\partial + \partial^\dagger$ with a scaling factor of $\alpha = N$. As a result of the reduced ancilla count and improved scaling factor, we expect the first approach to have better performance.

We will implement a $(\sqrt{N}, 2, 0)$ block-encoding of ∂_k^i

$$(I_N \otimes \langle 00|) V_{\partial_k^i} (I_N \otimes |00\rangle) = \frac{\partial_k^i}{\sqrt{N}} \tag{51}$$

with

$$\partial_k^i = \sum_{s_k \in S_k^i} \sum_{j=0}^{k} (-1)^j \left|s_k(\hat{v}_j)\right\rangle \langle s_k| \tag{52}$$

where $s_k(\hat{v}_j)$ denotes the Hamming-weight $k+1$ simplex s_k with the jth 1 value set to 0. We first observe that

$$\begin{aligned}
&X_{a_1} O_{m_k^i} \left(I_N \otimes |0\rangle_{a_1}\right) \\
&= X_{a_1} O_{m_k^i} \left(\sum_{s_k \in S_k^i} |s_k\rangle \langle s_k| + \sum_{x \notin S_k^i} |x\rangle \langle x| \right) \otimes |0\rangle_{a_1} \\
&= \left(\sum_{s_k \in S_k^i} |s_k\rangle \langle s_k| \right) \otimes |0\rangle_{a_1} + \left(\sum_{x \notin S_k^i} |x\rangle \langle x| \right) \otimes |1\rangle_{a_1} .
\end{aligned}$$

Similarly

$$\begin{aligned}
&\left(I_N \otimes \langle 0|_{a_2}\right) O_{m_{k-1}^i} X_{a_2} \\
&= \left(\sum_{s_{k-1} \in S_{k-1}^i} |s_{k-1}\rangle \langle s_{k-1}| + \sum_{x \notin S_{k-1}^i} |x\rangle \langle x| \right) \otimes \langle 0|_{a_2} O_{m_{k-1}^i} X_{a_2} \\
&= \left(\sum_{s_{k-1} \in S_{k-1}^i} |s_{k-1}\rangle \langle s_{k-1}| \right) \otimes \langle 0|_{a_2} + \left(\sum_{x \notin S_{k-1}^i} |x\rangle \langle x| \right) \otimes \langle 1|_{a_2} .
\end{aligned}$$

Sandwiching the circuit from Ref. [43] between these operations provides the resulting block-encoding of ∂_k^i. This approach has a gate depth of $\log(N)$ plus one call each to the membership oracles $O_{m_k^i}$ and $O_{m_{k-1}^i}$. We implement an equivalent block-encoding of ∂_{k+1}^j. In Table. 12 we compare the costs of the approach outlined above to the method used to block-encode ∂_k^i in Ref. [17].

C.3 Block encodings of subspace projectors

In this section we discuss how to implement the following block encodings:

- $V_{\Pi_{\text{Ker}}}$: A block encoding of Π_{Ker}, the projector onto the kernel of ∂_k^i.
- $V_{\Pi_{\text{Im}}}$: A block encoding of Π_{Im}, the projector onto the image of ∂_{k+1}^j.
- $V_{\Pi_{\text{Ker}(\partial_k^i) \cap \text{Im}(\partial_{k+1}^j)}}$: A block encoding of $\Pi_{\text{Ker} \cap \text{Im}}$, the projector onto $\text{Ker}(\partial_k^i) \cap \text{Im}(\partial_{k+1}^j)$.

	Here [43]	Ref [17]
Ancilla qubits	2	$\sim (\lceil \log(N) \rceil + 2\lceil \log(k) \rceil + 5)$
α	$\sqrt{N}$	Nk
Non-Clifford gates	$2(N-1)$ single-qubit rotations	$\mathcal{O}(N^2)$ Toffoli gates
Gate depth	$\mathcal{O}(\log(N))$	$\Omega(N)$
Calls to membership oracle	2	1

Table 12: A comparison of the approach outlined here for implementing a block-encoding of ∂_k^i in the direct mapping against the approach of Ref. [17].

- V_{Π_β}: A block encoding of $\Pi_{\text{Ker}} - \Pi_{\text{Ker}\cap\text{Im}}$, the projector onto $\text{Ker}(\partial_k^i) - \text{Ker}(\partial_k^i) \cap \text{Im}(\partial_{k+1}^j)$.

These will be constructed from QSVT circuits applied to the block encodings of ∂_k^i and ∂_{k+1}^j. We use the same approach for both the compact and direct mappings.

C.3.1 Projector onto kernel

The QSVT circuit for implementing $V_{\Pi_{\text{Ker}}}$ uses singular value threshold projectors [32], i.e., it (approximately) transforms the block-encoding of ∂_k^i into a block-encoding of the projector Π_{Ker} by mapping 0 singular values to 1 and non-zero singular values (that are at least $\Lambda_{\partial_k^i}$) to approximately 0. A technical detail is that we need to use a polynomial of even degree for singular value transformation [32] ensuring that the left and right singular vectors coincide, yielding an (approximate) orthonormal projector. Suitable definite-parity polynomials can be found in Refs. [32, 45, 80, 81]. We need to filter out singular values above a threshold $\Lambda_{\partial_k^i}$. Using the results of [32, Theorem 31] given an $(\alpha, m, 0)$ block encoding of an operator A, we can implement $V_{\Pi_{\text{Ker}(A)}}$, which is a $(1, m+1, \epsilon_k)$ block encoding of $\Pi_{\text{Ker}(A)}$. We require $\mathcal{O}\left(\frac{\alpha}{\Lambda}\log(\epsilon_k^{-1})\right)$ calls to an $(\alpha, m, 0)$ block encoding of A, and its inverse (where Λ is the gap between the zero and lowest non-zero singular values).

As a result, in the compact mapping we can implement a $(1, \log(N+1) + \log(k+1) + 2, \epsilon_k)$ block encoding of Π_{Ker} using $\mathcal{O}\left(\frac{\sqrt{(N+1)(k+1)}}{\Lambda_{\partial_k^i}}\log(\epsilon_k^{-1})\right)$ calls to $V_{\partial_k^i}$, and its inverse. We also require $\mathcal{O}\left(\frac{\sqrt{(N+1)(k+1)}}{\Lambda_{\partial_k^i}}\log(\epsilon_k^{-1})\right)$ calls to a $(\log(N+1) + \log(k+1) + 1)$-controlled-Toffoli gate (and the same number of calls to a $(\log(k+1)+1)$-controlled-Toffoli gate), which can be implemented in $\mathcal{O}(\log(N+1) + \log(k+1) + 1)$ depth.

For the direct mapping, we can implement a $(1, 3, \epsilon_k)$ block encoding of Π_{Ker} using $\mathcal{O}\left(\frac{\sqrt{N}}{\Lambda_{\partial_k^i}}\log(\epsilon_k^{-1})\right)$ calls to $V_{\partial_k^i}$, and its inverse, as well as a similar number of calls to a Toffoli gate.

Note that the block-encoding of ∂_k^i makes calls to the membership oracle, which dominates the cost of the QSVT circuit, compared to the multicontrol Toffoli gates. As a result, we ignore the costs of the multicontrol Toffoli gates going forwards.

C.3.2 Projector onto image

To implement a projector onto the image of a matrix A, consider its singular value decomposition $A = \sum_i \sigma_i |L_i\rangle \langle R_i|$. A projector onto the image of A is given by $\Pi_{\text{Im}(A)} = \sum_{j;\sigma_j>0} |L_j\rangle \langle L_j|$. We can implement a block encoding of this projector by using QSVT to apply an even polynomial approximating a threshold function to the singular values of the encoded operator $A^\dagger$ (which can be implemented using $V_A^\dagger$, when V_A is a block encoding of A). This ensures that all singular values above the threshold Λ/α are set to 1, and all values below are set to zero. The use of an even polynomial ensures that we only keep the left singular vectors of A [32]. We can implement $V_{\Pi_{\text{Im}(A)}}$, which is a $(1, m+1, \epsilon_i)$ block

encoding of $\Pi_{\mathrm{Im}(A)}$ using $\mathcal{O}\left(\frac{\alpha}{\Lambda}\log(\epsilon_i^{-1})\right)$ calls to an $(\alpha, m, 0)$ block encoding of A, and its inverse, and the same number of calls to m-controlled-Toffoli gates.

As a result, in the compact mapping we can implement a $(1, \log(N+1)+\log(k+2)+2, \epsilon_i)$ block encoding of Π_{Im} using $\mathcal{O}\left(\frac{\sqrt{(N+1)(k+2)}}{\Lambda_{\partial_{k+1}^j}}\log(\epsilon_i^{-1})\right)$ calls to $V_{\partial_{k+1}^j}$, and its inverse, and the same number of calls to $(\log(N+1)+\log(k+2)+1)$- and $(\log(k+2)+1)$-controlled Toffoli gates.

For the direct mapping, we can implement a $(1, 3, \epsilon_i)$ block encoding of Π_{Im} using $\mathcal{O}\left(\frac{\sqrt{N}}{\Lambda_{\partial_{k+1}^j}}\log(\epsilon_i^{-1})\right)$ calls to $V_{\partial_{k+1}^j}$, and its inverse, and the same number of calls to additional Toffoli gates.

As described above, we ignore the costs of the multicontrol Toffoli gates going forwards.

C.3.3 Projector onto kernel and image

Given the above methods for implementing $V_{\Pi_{\mathrm{Ker}}}$ and $V_{\Pi_{\mathrm{Im}}}$, we can implement $V_{\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}}$. We cannot simply take a product of the two projectors, as in general they do not commute with each other (because of new simplices that enter the complex at scale j). We can implement $V_{\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}}$ by first using a product of block encodings [32] to obtain a product of the projectors $\Pi_{\mathrm{Ker}}\cdot\Pi_{\mathrm{Im}}$. We can use Ref. [32, Lemma 53], which states that:
If U is an (α, a, δ) block encoding of A and V is a (β, b, ϵ) block encoding of B, then $(I_b\otimes U)(I_a\otimes V)$ is an $(\alpha\beta, a+b, \alpha\epsilon+\beta\delta)$ block encoding of AB.

As a result, in the compact mapping we can implement a

$$(1, 2\log(N+1)+\log(k+1)+\log(k+2)+4, \epsilon_k+\epsilon_i) \tag{53}$$

block encoding of $\Pi_{\mathrm{Ker}}\Pi_{\mathrm{Im}}$ using one call each to of $V_{\Pi_{\mathrm{Ker}}}$ and $V_{\Pi_{\mathrm{Im}}}$.

In the direct mapping we can implement a

$$(1, 6, \epsilon_k+\epsilon_i) \tag{54}$$

block encoding of $\Pi_{\mathrm{Ker}}\Pi_{\mathrm{Im}}$ using one call each to of $V_{\Pi_{\mathrm{Ker}}}$ and $V_{\Pi_{\mathrm{Im}}}$.

We then apply QSVT to this block encoding. We apply an even function that sends all non-unity singular values to zero. In reality, we can only send singular values below a threshold $(1-\Lambda_{\Pi\Pi})$ to zero. We require $\mathcal{O}\left(\Lambda_{\Pi\Pi}^{-0.5}\log\left(\epsilon_p^{-1}\right)\right)$ (see [39, Lemma 35] for an explanation of the quadratic improvement) calls to $V_{\Pi_{\mathrm{Ker}}}$ and $V_{\Pi_{\mathrm{Im}}}$ (and their inverses) to implement $V_{\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}}$ in this way. The error in the above block encoding is scaled according to the robustness of QSVT [32, Lemma 22], by $\mathcal{O}\left(4\Lambda_{\Pi\Pi}^{-0.5}\log\left(\epsilon_p^{-1}\right)\sqrt{\epsilon_k+\epsilon_i}\right)$. The QSVT circuit requires $\mathcal{O}\left(\Lambda_{\Pi\Pi}^{-0.5}\log\left(\epsilon_p^{-1}\right)\right)$ calls to either $(2\log(N+1)+\log(k+1)+\log(k+2)+4)$-controlled-Toffoli gates (compact) or 6-controlled-Toffoli gates (direct). We neglect the cost of these multicontrol Toffoli gates going forwards, as they are dominated by the cost of $V_{\Pi_{\mathrm{Ker}}}$ and $V_{\Pi_{\mathrm{Im}}}$.

We can prepare the block encoding V_{Π_β} of $\Pi_{\mathrm{Ker}(\partial_k^i)}-\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}$ using a linear combination of the block encodings $V_{\Pi_{\mathrm{Ker}}}$ and $V_{\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}}$ [39, Lemma 52]. This adds a control to each of the operations $V_{\Pi_{\mathrm{Ker}}}$ and $V_{\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}}$, which can be achieved by adding a control qubit to each of the QSVT rotations in their constituent circuits. The overall cost to implement V_{Π_β} of $\Pi_{\mathrm{Ker}(\partial_k^i)}-\Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}$ is shown in Table 13.

C.4 State preparation unitary V_ψ

In this section we construct the state preparation unitary V_ψ that approximately prepares the state $|\psi_{S_k^i}\rangle$, the purification of the maximally mixed state over k-simplices in the complex at scale i. Our approach uses fixed-point amplitude amplification, analyzed from a QSVT perspective [32].

	(α, m, ϵ)	Costs
Compact	$\left(2, 2\log(N+1)+\log(k+1)+\log(k+2)+6,\ \frac{8}{\Lambda_{\Pi\Pi}^{0.5}}\log\left(\epsilon_p^{-1}\right)\sqrt{\epsilon_k+\epsilon_i}+\epsilon_p+\epsilon_k\right)$	$\mathcal{O}\left(\frac{1}{\Lambda_{\Pi\Pi}^{0.5}}\log\left(\frac{1}{\epsilon_p}\right)\right)\times$ $V_{\Pi_{\mathrm{Ker}}}, V_{\Pi_{\mathrm{Ker}}}^\dagger, V_{\Pi_{\mathrm{Im}}}, V_{\Pi_{\mathrm{Im}}}^\dagger$
Direct	$\left(2, 8, \frac{8}{\Lambda_{\Pi\Pi}^{0.5}}\log\left(\epsilon_p^{-1}\right)\sqrt{\epsilon_k+\epsilon_i}+\epsilon_p+\epsilon_k\right)$	

Table 13: A summary of the costs to implement the block encoding V_{Π_β} of $\Pi_{\mathrm{Ker}(\partial_k^i)} - \Pi_{\mathrm{Ker}(\partial_k^i)\cap\mathrm{Im}(\partial_{k+1}^j)}$.

C.4.1 Preparing purification of maximally mixed state over all possible k-simplices

In this section we consider unitary circuits U_{uni} that prepare the purification of the maximally mixed state over all $\binom{N}{k+1}$ possible k-simplices in the complex. We will discuss how to implement unitaries that first prepare an equal superposition over all possible k-simplices in the complex. We can then introduce a second register of equal size, and apply CNOT gates between the corresponding qubits in each register, to generate the purification.

The direct encoding stores k-simplices as Hamming weight $(k+1)$ computational basis states of N qubits. An equally weighted superposition of states with fixed Hamming weight is known as a Dicke state. We can prepare Dicke states using the approaches in Refs. [46, 47], the most efficient of which requires $\mathcal{O}(k\log(N/k))$ depth.

Preparing an equal superposition of k-simplices in the compact encoding proceeds as follows. We will consider $N+1=2^m$ for some integer m to simplify the analysis. The uniform superposition over k-simplices built from N datapoints is

$$\frac{1}{\sqrt{\binom{N}{k+1}}}\sum_{s_k}|s_k\rangle = \frac{1}{\sqrt{\binom{N}{k+1}}}\sum_{V_1=1}^{[N-k]}\sum_{V_2>V_1}^{[N-k+1]}\cdots\sum_{V_{k+1}>V_k}^{N}|V_1\rangle|V_2\rangle\ldots|V_{k+1}\rangle \tag{55}$$

where each register acts on $m=\log(N+1)$ qubits. We can prepare this state by first placing each register in an equal superposition of N states

$$\left(\frac{1}{\sqrt{N}}\sum_{V_i=1}^{N}|V_i\rangle\right)^{\otimes(k+1)}. \tag{56}$$

The dimension of the state above is N^{k+1}. Preparing the superpositions on each register can be done (for each register in parallel) using $\mathcal{O}(\log(N))$ gates [75] (see also Refs. [82–84]). We need to isolate the components of this state that are permutations of our desired orderings (i.e. we need to remove branches of the superposition with repeated vertices)

$$\frac{1}{\sqrt{(k+1)!\binom{N}{k+1}}}\sum_{V_1\neq V_2\ldots\neq V_{k+1}}^{N}|V_1\rangle|V_2\rangle\ldots|V_{k+1}\rangle \tag{57}$$

We check all pairs of registers using k parallel rounds of $(k+1)/2$ equality checks. Each in-place equality check uses $\log(N+1)$ CNOT gates followed by a $\log(N+1)$-controlled Toffoli gate targeting an ancilla qubit. The gate depth can be reduced by introducing $\log(N+1)$ ancilla qubits (for each register pairing), and using a tree structure with Toffoli depth $\log\log(N)$ to replace the $\log(N+1)$-controlled Toffoli gate. We then use a $(k+1)/2$-controlled Toffoli to flag any failed equality check in that round. Once again we can replace the $(k+1)/2$-controlled Toffoli with $(k+1)/2$ ancilla qubits and a tree structure with with Toffoli depth $\mathcal{O}(\log(k))$. After all the rounds we require a final k-controlled Toffoli to trigger the main flag qubit (which can again be replaced by ancilla qubits and a tree structure). Overall the repeated vertex detection requires $\mathcal{O}(k(\log\log(N)+\log(k)))$ gates and $\mathcal{O}(k\log(N))$ ancilla qubits.

The dimension of the desired 'permutation state' is $N \times ... \times (N-k) = \frac{N!}{(N-k-1)!}$ so the probability of not seeing a duplicate vertex is $\prod_{j=0}^{k} \frac{N-j}{N} = \prod_{j=1}^{k}(1-\frac{j}{N}) = \exp\Big(\sum_{j=1}^{k} \ln(1-\frac{j}{N})\Big)$. We can use exact amplitude amplification, because we can classically exactly compute the above probability of not getting repeated vertices. As long as $j \leq k+1 \leq \frac{N}{2}$ we have $\ln(1-\frac{j}{N}) \geq -\frac{2j}{N}$ and the above probability can be lower bounded by $\exp\Big(-\sum_{j=1}^{k} \frac{2j}{N}\Big) = \exp(-k(k+1)/N)$, upper bounding the required number of amplitude amplification rounds by $\mathcal{O}\Big(\exp\Big(2\binom{k+1}{2}/N\Big)\Big)$.

Once the permutation state is prepared it can be sorted into the correct order using a reversible quantum sorting network, as described in Sec. 6.3.1 of Ref. [48]. The sorting network requires $\mathcal{O}(k\log(k)\log\log(N))$ depth and $\mathcal{O}(k(\log(k)+\log(N))) = \mathcal{O}(k\log(N))$ additional ancilla qubits. At the end of the sorting network, the ancilla register is unentangled with the desired state, and so can be treated as part of the purifying register. Therefore, assuming that $k \leq \frac{N}{\log(N)}$ (otherwise we would advise using the direct encoding anyway), the overall complexity of preparing the equal superposition over simplices in the compact mapping is

$$\mathcal{O}\left((\log(N)+k\log(k)+k\log\log(N))\exp\left(2\binom{k+1}{2}/N\right)+k\log(k)\log\log(N)\right) \tag{58}$$

and requires

$$\mathcal{O}(k\log(N)) \tag{59}$$

additional ancilla qubits.

C.4.2 Preparing the purified maximally mixed state over k-simplices in the complex via fixed-point amplitude amplification

Given the above unitary circuits that prepare the purification of the maximally mixed state over all possible k-simplices in the complex, we can use fixed point amplitude amplification (implemented using QSVT) to generate V_ψ. A thorough exposition of this technique is given in Ref. [81, Section 3].

Our presentation makes use of a generalization of block-encodings, known as projected unitary encodings [32]. The necessary details are captured by the following Definitions and Lemma, which are less formal restatements of the results of Ref. [32].

Definition 3. *We say an $n+m$ qubit operator U is an (α, m, ϵ) projected unitary encoding of the n qubit operator A if $||A - \alpha\Pi_L U \Pi_R|| \leq \epsilon$, where Π_L, Π_R are orthogonal projectors.*

Definition 4. *We define a $C_\Pi NOT$ gate as*

$$\mathrm{C}_\Pi\mathrm{NOT} = (I-\Pi)\otimes I + \Pi \otimes X. \tag{60}$$

For example, if $\Pi = |1\rangle\langle 1|$, $C_\Pi NOT$ is a regular CNOT gate, while if $\Pi = |11\rangle\langle 11|$, $C_\Pi NOT$ is a Toffoli gate.

Assume we are given an (α, m, ϵ) projected unitary encoding U of an operator A, which has singular value decomposition $A/\alpha = W\Sigma V^\dagger$. Assume we are also given the associated $\mathrm{C}_{\Pi_L}\mathrm{NOT}$ and $\mathrm{C}_{\Pi_R}\mathrm{NOT}$ gates, as well as a degree d real polynomial $P(x)$, $|P(x)| \leq 1$ for $x \in [-1,1]$. QSVT provides the following Lemma

Lemma 1. *We can implement a projected unitary encoding U_o for a degree d odd polynomial, such that $\Pi_L U_o \Pi_R = P_o(A/\alpha) := W P_o(\Sigma) V^\dagger$ (and equivalently U_e for a degree d even polynomial such that $\Pi_R U_e \Pi_R = P_e(A/\alpha) := V P_e(\Sigma) V^\dagger$) using a QSVT circuit. The QSVT circuit makes d calls to U, $U^\dagger$, $2d$ calls to $C_{\Pi_L}NOT$, $C_{\Pi_R}NOT$, and uses $\mathcal{O}(d)$ other single- and two-qubit gates.*

We define the following operations: U_{uni} (as above) prepares the purified maximally mixed state over all possible k-simplices, $\Pi_0 = |\bar{0}\rangle\langle\bar{0}|$, $\Pi_{\psi_k^i} = |\psi_{S_k^i}\rangle\langle\psi_{S_k^i}|$. Recall that

$$|\psi_{S_k^i}\rangle := \frac{1}{\sqrt{|S_k^i|}} \sum_{s_k \in S_k^i} |s_k\rangle |s_k\rangle . \tag{61}$$

By analogy with Grover's algorithm, we note that the membership oracle $O_{m_k^i}$ acts as a $C_{\Pi_{\psi_k^i}} NOT$ gate, with $\Pi_{\psi_k^i} = |\psi_{S_k^i}\rangle\langle\psi_{S_k^i}|$, when we restrict to the 2D subspace spanned by $|\psi_{S_k^i}\rangle, |\psi_{S_k^i}^{\perp}\rangle$, which we remain in during the course of this subroutine. We then see that

$$\left\| \Pi_{\psi_k^i} U_{\text{uni}} \Pi_0 - \sqrt{\frac{|S_k^i|}{\binom{N}{k+1}}} |\psi_{S_k^i}\rangle\langle\bar{0}| \right\| = 0 \tag{62}$$

i.e. U_{uni} provides a $(1,0,0)$ projected unitary encoding of $\sqrt{\frac{|S_k^i|}{\binom{N}{k+1}}} |\psi_{S_k^i}\rangle\langle\bar{0}|$. We can use QSVT to map the singular value $\sqrt{\frac{|S_k^i|}{\binom{N}{k+1}}}$ to 1. A layer of the resulting QSVT circuit in shown in Fig. 3. In this circuit, the membership oracle fulfils the role of the $\text{C}_{\Pi_L}\text{NOT}$ gates.

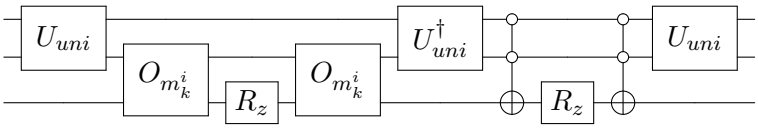


Figure 3: A layer of the QSVT implementation of fixed-point amplitude amplification used to implement the state preparation unitary V_ψ.

The resulting QSVT circuit V_ψ is a

$$(1, 1, \epsilon_\psi) \tag{63}$$

projected unitary encoding of $|\psi_{S_k^i}\rangle\langle\bar{0}|$, and uses $\mathcal{O}\left(\sqrt{\frac{\binom{N}{k+1}}{|S_k^i|}} \log(\epsilon_\psi^{-1})\right)$ calls to $O_{m_k^i}$, U_{uni}, $U_{\text{uni}}^{\dagger}$, and Toffoli gates controlled on all register qubits of the purification ($(2(k+1)\log(N+1))$ qubits for the compact mapping, $2N$ qubits for the direct mapping).

D Determining the overall complexity of the algorithm

In this section we determine the overall complexity of our algorithm. We invoke Corollary 3.1, which we restate here:

To estimate $\beta_k^{i,j}$ to additive error Δ with success probability $\geq 1-\eta$, we solve two instances of normalized projector rank estimation. The first encodes the value of $|S_k^i|/\binom{N}{k+1}$, and the second encodes the value of $\beta_k^{i,j}/|S_k^i|$. Each instance uses $\mathcal{O}(\log(\eta^{-1}))$ repetitions of a quantum circuit that makes $\mathcal{O}(\delta_x^{-1})$ calls to the state preparation unitaries V_{ψ_x} (acting on b_x qubits) and the $(\alpha_x, a_x, \delta_x/4)$ block-encodings V_{Π_x} of projector Π_x shown in Table 14. Each circuit also uses $\mathcal{O}\left((a_x+b_x)\frac{\alpha_x}{\delta_x}\right)$ additional two-qubit gates.

We will make use of the following facts and assumptions throughout:

- Instance 2 is more costly than instance 1, and hence we use the cost of instance 2 to bound the asymptotic cost of the algorithm.
- We assume the cost of calls to membership oracles in the algorithm have a cost equal to $O_{m_k^i}$ (for $k-1, k, k+1$ and i, j) (this is true for the direct mapping, and asymptotically true for the compact mapping).

Instance x	1	2
Estimates	$X := \sqrt{\lvert S_k^i\rvert / \binom{N}{k+1}}$	$Y := \sqrt{\beta_k^{i,j} / \lvert S_k^i\rvert}$
δ_x	$\frac{\Delta}{4\beta_k^{i,j}}\sqrt{\frac{\lvert S_k^i\rvert}{\binom{N}{k+1}}}$	$\frac{\Delta}{4\sqrt{\lvert S_k^i\rvert \beta_k^{i,j}}}$
Π_x	$\Pi_{S_k^i}$: Projects onto k-simplices in complex at scale i	$\Pi_{\mathrm{Ker}} - \Pi_{\mathrm{Ker}\cap\mathrm{Im}}$: Projects onto $\mathrm{Ker}(\partial_k^i) - \mathrm{Ker}(\partial_k^i) \cap \mathrm{Im}(\partial_{k+1}^j)$
$\lvert\psi_x\rangle$	$\lvert\psi_s\rangle$: Uniform superposition of all possible k-simplices	$\lvert\psi_{S_k^i}\rangle$: Purification of maximally mixed state over all k-simplices in complex at scale i

Table 14: Details of the error parameters and block encoded operators for each of the instances of normalized projector rank estimation used to estimate the persistent Betti number to additive error Δ using the algorithm of Theorem 2.

- We assume the inverse of a block-encoding unitary $V^\dagger$ has the same cost as the unitary V.

The cost of the quantum circuit is then given by:

$$\mathcal{O}\left(\frac{\alpha_2}{\delta_2} \times \left(V_\psi + V_{\Pi_\beta}\right)\right). \tag{64}$$

In the following subsections, we will determine these costs in terms of the building blocks introduced in the preceding sections. We will do this for both the direct mapping, and the compact mapping with a slow-load lookup table. We will not treat the compact mapping with a fast-load lookup table explicitly, as it is a small modification of the slow-load results, and the large increase in ancilla qubits comes at the expense of a modest improvement in the gate complexity.

D.1 Direct mapping

We focus first on the direct mapping. We have the following costs:

- $O_{m_k^i}$: $\mathcal{O}(N\log(N))$ depth and $\mathcal{O}(N)$ additional qubits.
- V_ψ : uses $\mathcal{O}\left(\sqrt{\frac{\binom{N}{k+1}}{\lvert S_k^i\rvert}}\log\left(\frac{1}{\epsilon_\psi}\right)\right) \times [U_{\mathrm{Uni}} + O_{m_k^i}]$.
  - U_{Uni} : $\mathcal{O}(k\log(N))$ depth and no additional qubits.
- V_{Π_β} : $\left(2, 8, \epsilon_p + \epsilon_k + \frac{8}{\Lambda_{\mathrm{III}}^{0.5}}\log\left(\epsilon_p^{-1}\right)\sqrt{\epsilon_k + \epsilon_i}\right)$ block-encoding, using $\mathcal{O}\left(\frac{1}{\Lambda_{\mathrm{III}}^{0.5}}\frac{\sqrt{N}}{\mathrm{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\log\left(\frac{1}{\epsilon_p}\right)\log\left(\frac{1}{\epsilon_{i/k}}\right)\right) \times [V_{\partial_k^i}, V_{\partial_{k+1}^j}]$.
  - $V_{\partial_k^i}, V_{\partial_{k+1}^j}$: $\mathcal{O}(\log(N)) + O_{m_k^i}$ depth, and only the additional qubits required for $O_{m_k^i}$.

We refer to the error $\epsilon_{i/k}$ because as we shall show below, ϵ_i and ϵ_k can be chosen to be the same.

Substituting these costs into the overall algorithmic cost and dropping subleading terms (U_{Uni}, $\log(N)$ depth in V_∂) gives:

$$\mathcal{O}\left(\frac{N\log(N)}{\delta_2} \times \left(\sqrt{\frac{\binom{N}{k+1}}{\lvert S_k^i\rvert}}\log\left(\frac{1}{\epsilon_\psi}\right) + \frac{1}{\Lambda_{\mathrm{III}}^{0.5}}\frac{\sqrt{N}}{\mathrm{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\log\left(\frac{1}{\epsilon_p}\right)\log\left(\frac{1}{\epsilon_{i/k}}\right)\right)\right)$$

We must determine suitable values for the errors $\epsilon_\psi, \epsilon_p, \epsilon_{i/k}$. As stated in Corollary 3.1, the error in V_{Π_β} must be equal to $\delta_2/4$. Hence

$$\epsilon_p + \epsilon_k + \frac{8}{\Lambda_{\Pi\Pi}^{0.5}} \log\left(\epsilon_p^{-1}\right)\sqrt{\epsilon_k + \epsilon_i} \sim \delta_2. \tag{65}$$

We can thus choose

$$\epsilon_p \sim \delta_2 \tag{66}$$

$$\epsilon_{i/k} \sim \frac{\delta_2^2 \Lambda_{\Pi\Pi}}{\log^2\left(\frac{1}{\delta_2}\right)} \tag{67}$$

Similarly $\epsilon_\psi = \delta_2/4$. Substituting these values into the cost of the algorithm, and hiding constant terms using the big-$\mathcal{O}$ notation yields

$$\mathcal{O}\left(\frac{N\log(N)}{\delta_2}\log\left(\frac{1}{\delta_2}\right) \times \left(\sqrt{\frac{\binom{N}{k+1}}{|S_k^i|}} + \frac{\sqrt{N}}{\Lambda_{\Pi\Pi}^{0.5}\mathrm{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\log\left(\frac{1}{\delta_2^2\Lambda_{\Pi\Pi}}\right)\right)\right). \tag{68}$$

Substituting in the value of δ_2 yields

$$\mathcal{O}\left(\frac{N\log(N)\sqrt{\beta_k^{i,j}}}{\Delta}\log\left(\frac{\sqrt{\beta_k^{i,j}|S_k^i|}}{\Delta}\right) \times \left(\sqrt{\binom{N}{k+1}} + \frac{\sqrt{N|S_k^i|}}{\Lambda_{\Pi\Pi}^{0.5}\mathrm{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\log\left(\frac{\beta_k^{i,j}|S_k^i|}{\Delta^2\Lambda_{\Pi\Pi}}\right)\right)\right). \tag{69}$$

We consider $|S_k^i| \sim \binom{N}{k+1}$, as this is the regime in which classical algorithms are least efficient. The resulting complexity is:

$$\mathcal{O}\left(\frac{N^{3/2}\log(N)\sqrt{\binom{N}{k+1}\beta_k^{i,j}}}{\Delta\Lambda_{\Pi\Pi}^{0.5}\mathrm{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\log\left(\frac{\sqrt{\binom{N}{k+1}\beta_k^{i,j}}}{\Delta}\right)\log\left(\frac{\binom{N}{k+1}\beta_k^{i,j}}{\Delta^2\Lambda_{\Pi\Pi}}\right)\right) \tag{70}$$

depth. We require

$$\tilde{\mathcal{O}}\left(\log\left(\frac{1}{\eta}\right)\right) \tag{71}$$

incoherent repetitions of this circuit. Hence, the overall complexity is

$$\tilde{\mathcal{O}}\left(\frac{N^{3/2}\sqrt{\binom{N}{k+1}\beta_k^{i,j}}}{\Delta\Lambda_{\Pi\Pi}^{0.5}\mathrm{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\right). \tag{72}$$

The quantum circuit used acts on $\mathcal{O}(N)$ qubits.

D.2 Compact mapping

For the compact mapping using a slow-load lookup table, we have the following costs:

- $O_{m_k^i}$: $\mathcal{O}(N + k(\log(d)\log(b_d) + b_d))$ depth and $\mathcal{O}\big(k\log(N) + kdb_d^2\big)$ additional qubits.
- V_ψ : uses $\mathcal{O}\left(\sqrt{\frac{\binom{N}{k+1}}{|S_k^i|}}\log\left(\frac{1}{\epsilon_\psi}\right)\right) \times [U_{\mathrm{Uni}} + O_{m_k^i}]$.

- U_{Uni} : $\mathcal{O}(\log(N)) + k\log(k)\log\log(N)$ depth (when $k \in \mathcal{O}(\sqrt{N})$) and $\mathcal{O}(k\log(N))$ additional qubits.

- V_{Π_β} : $\left(2, \log((N+1)^2(k+1)(k+2)) + 6, \epsilon_p + \epsilon_k + \frac{8}{\Lambda_{\Pi\Pi\Pi}^{0.5}}\log\left(\epsilon_p^{-1}\right)\sqrt{\epsilon_k + \epsilon_i}\right)$ block-encoding, using $\mathcal{O}\left(\frac{1}{\Lambda_{\Pi\Pi\Pi}^{0.5}}\frac{\sqrt{Nk}}{\text{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\log\left(\frac{1}{\epsilon_p}\right)\log\left(\frac{1}{\epsilon_{i/k}}\right)\right) \times [V_{\partial_k^i}, V_{\partial_{k+1}^j}]$.

  - $V_{\partial_k^i}, V_{\partial_{k+1}^j}$: $\mathcal{O}(k\log\log(N)) + O_{m_k^i}$ depth, and $\mathcal{O}(\log(N))$ additional qubits, plus those required for $O_{m_k^i}$.

We refer to the error $\epsilon_{i/k}$ because as we shall show below, ϵ_i and ϵ_k can be chosen to be the same.

We work in regime with $\mathcal{O}(k\log(k)\log\log(N)) \in \mathcal{O}(N)$. Substituting these costs into the overall algorithmic cost and dropping subleading terms gives:

$$\mathcal{O}\left(\frac{N + k(\log(d)\log(b_d) + b_d)}{\delta_2} \times \left(\sqrt{\frac{\binom{N}{k+1}}{|S_k^i|}}\log\left(\frac{1}{\epsilon_\psi}\right) + \frac{1}{\Lambda_{\Pi\Pi\Pi}^{0.5}}\frac{\sqrt{Nk}}{\text{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\log\left(\frac{1}{\epsilon_p}\right)\log\left(\frac{1}{\epsilon_{i/k}}\right)\right)\right)$$

Repeating the error analysis performed for the direct mapping,

$$\epsilon_p \sim \delta_2 \tag{73}$$

$$\epsilon_{i/k} \sim \frac{\delta_2^2 \Lambda_{\Pi\Pi\Pi}}{\log^2\left(\frac{1}{\delta_2}\right)} \tag{74}$$

$$\epsilon_\psi \sim \delta_2 \tag{75}$$

Substituting these values into the cost of the algorithm, and hiding constant terms using the big-$\mathcal{O}$ notation yields

$$\mathcal{O}\left(\frac{N + k(\log(d)\log(b_d) + b_d)}{\delta_2}\log\left(\frac{1}{\delta_2}\right) \times \left(\sqrt{\frac{\binom{N}{k+1}}{|S_k^i|}} + \frac{\sqrt{Nk}}{\Lambda_{\Pi\Pi\Pi}^{0.5}\text{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\log\left(\frac{1}{\delta_2^2\Lambda_{\Pi\Pi\Pi}}\right)\right)\right). \tag{76}$$

Substituting in the value of δ_2 yields

$$\mathcal{O}\left(\frac{(N + k(\log(d)\log(b_d) + b_d))\sqrt{\beta_k^{i,j}|S_k^i|}}{\Delta}\log\left(\frac{\sqrt{\beta_k^{i,j}|S_k^i|}}{\Delta}\right) \times \left(\sqrt{\frac{\binom{N}{k+1}}{|S_k^i|}} + \frac{\sqrt{Nk}}{\Lambda_{\Pi\Pi\Pi}^{0.5}\text{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\log\left(\frac{\beta_k^{i,j}|S_k^i|}{\Delta^2\Lambda_{\Pi\Pi\Pi}}\right)\right)\right) \tag{77}$$

We consider $|S_k^i| \sim \binom{N}{k+1}$, as this is the regime in which classical algorithms are least efficient. Noting that the Johnson-Lidenstrauss lemma ensures we can choose $d \in \mathcal{O}(\log(N)/\epsilon_{JL}^2)$, we assume that $\mathcal{O}(k(\log(d)\log(b_d) + b_d)) \in \mathcal{O}(N)$. The resulting complexity is:

$$\mathcal{O}\left(\frac{N^{3/2}\sqrt{k}\sqrt{\binom{N}{k+1}\beta_k^{i,j}}}{\Delta\Lambda_{\Pi\Pi\Pi}^{0.5}\text{Min}\left(\Lambda_{\partial_k^i}, \Lambda_{\partial_{k+1}^j}\right)}\log\left(\frac{\sqrt{\binom{N}{k+1}\beta_k^{i,j}}}{\Delta}\right)\log\left(\frac{\binom{N}{k+1}\beta_k^{i,j}}{\Delta^2\Lambda_{\Pi\Pi\Pi}}\right)\right) \tag{78}$$

depth. We require

$$\tilde{\mathcal{O}}\left(\log\left(\frac{1}{\eta}\right)\right) \tag{79}$$

incoherent repetitions of this circuit. Hence, the overall complexity (circuit depth) is

$$\tilde{\mathcal{O}}\left(\frac{N^{3/2}\sqrt{k\beta_k^{i,j}\binom{N}{k+1}}}{\Delta\Lambda_{\text{III}}^{0.5}\text{Min}\left(\Lambda_{\partial_k^i},\Lambda_{\partial_{k+1}^j}\right)}\right). \tag{80}$$

The quantum circuit used acts on $\mathcal{O}(k\log(N))$ qubits.

E Nuances of persistent Betti numbers

E.1 Performing a change of basis

As discussed in the main text, there are a number of nuances associated with instantiating operators that encode persistent Betti numbers. In this work, we compute $\beta_k^{i,j}$ as

$$\beta_k^{i,j} = \dim\Big(\text{Ker}(\partial_k^i)\Big) - \dim\Big(\text{Ker}(\partial_k^i)\cap\text{Im}(\partial_{k+1}^j)\Big). \tag{81}$$

Nevertheless, there are other ways this problem can be formulated. For example, the task of computing persistent Betti numbers has been framed in terms of a persistent combinatorial Laplacian [58, 59]. This is achieved by first defining a subgroup $C_{k+1}^{i,j}(S^j)$ containing $(k+1)$-chains in the complex at scale j whose images under the boundary operator ∂_{k+1}^j are k-chains at scale i. Formally,

$$C_{k+1}^{i,j}(S^j) = \{c\in C_{k+1}(S^j) : \partial_{k+1}^j(c)\in C_k(S^i)\}. \tag{82}$$

Intuitively, the chains in $C_{k+1}^{i,j}(S^j)$ are mapped to k-cycles in $C_k(S^i)$, which can be divided into the k-boundaries already present in $C_k(S^i)$, and a subset of the k-holes in $C_k(S^i)$. In other words, the additional $(k+1)$-chains in $C_{k+1}^{i,j}(S^j)$ (beyond those already present in $C_{k+1}(S^i)$) have the action of filling-in k-holes in $C_k(S^i)$. Defining $\partial_{k+1}^{i,j}$ as ∂_{k+1} restricted to the chains (not just the simplices) in $C_{k+1}^{i,j}(S^j)$ it follows that [59]

$$\text{Im}(\partial_{k+1}^{i,j}) \cong \text{Ker}(\partial_k^i)\cap\text{Im}(\partial_{k+1}^j). \tag{83}$$

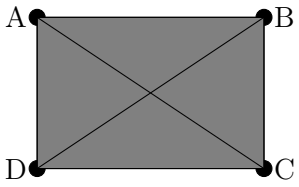
Moving from the simplex basis to the chain basis is the main subject of this section.

As discussed in Appendix A, $\beta_k^{i,j} = \dim\Big(\text{Ker}(\Delta_k^{i,j})\Big)$. The persistent combinatorial Laplacian matrix $\Delta_k^{i,j}$ can be built from the operator $\partial_{k+1}^{i,j}$ (c.f. Eq. (23)). Expressing $\partial_{k+1}^{i,j}$ in terms of matrices is more complex than expressing ∂_k^i, as discussed in Ref. [59], and as we shall show below. This is because the $(k+1)$-simplices in S_{k+1}^j do not form an appropriate basis for the group $C_{k+1}^{i,j}(S^j)$. Even if a chain $\sigma_1+\sigma_2\in C_{k+1}^{i,j}(S^j)$ (such that $\partial_{k+1}^j(\sigma_1+\sigma_2)\in C_k(S^i)$), it can be the case that $\partial_{k+1}^j(\sigma_1),\partial_{k+1}^j(\sigma_2)\notin C_k(S^i)$. The required change of basis adds complexity to both the classical and quantum algorithms for persistent Betti numbers. We elaborate more on this point below.

In this Appendix we provide a worked example for constructing the restricted boundary operator $\partial_{k+1}^{i,j}$, in order to build intuition for the increased complexity. Below, we show a pair of simplicial complexes at different scales i and j. Clearly there are zero 1-holes that persist from scale i to scale j, as the hole is filled by the new 2-simplices at scale j.

$$S_0^i = \{A, B, C, D\}$$
$$S_1^i = \{AB, BC, CD, AD\}$$
$$S_2^i = \{\}$$

$$S_0^j = \{A, B, C, D\}$$
$$S_1^j = \{AB, BC, CD, AD, AC, BD\}$$
$$S_2^j = \{ABC, ACD, ABD, BCD\}$$



We are interested in the elements of the restricted chain group

$$C_{k+1}^{i,j}(S^j) = \{c \in C_{k+1}(S^j) : \partial_{k+1}^j(c) \in C_k(S^i)\}. \tag{84}$$

In this example, it is straightforward to find the elements of this group manually. First, observe that

$$\begin{aligned} \partial_2^j[ABC] &= BC - AC + AB \\ \partial_2^j[ACD] &= CD - AD + AC \\ \partial_2^j[ABD] &= BD - AD + AB \\ \partial_2^j[BCD] &= CD - BD + BC. \end{aligned} \tag{85}$$

The 1-chains AC and BD are present at scale j, but are not present at scale i. Consequently, we need to take linear combinations of the 2-simplices in j that will cancel out these errant 1-chains to form the elements of $C_2^{i,j}(S^j)$. These are given by

$$\begin{aligned} c_1 &= ABC + ACD; \quad \partial_2^j[c_1] = AB + BC + CD - AD \\ c_2 &= ABD + BCD; \quad \partial_2^j[c_2] = AB + BC + CD - AD, \end{aligned} \tag{86}$$

or any linear combination of these chains. We see that while $c_1, c_2 \in C_2^{i,j}(S^j)$, their constituent simplices are not, as mentioned above. This is why we must represent $\partial_2^{i,j}$ in the basis of elements of $C_2^{i,j}(S^j)$, rather than in the basis of simplices. We can see that the dimension of $\partial_2^{i,j}$ is two, and its rank is one. The persistent Betti number is given by

$$\begin{aligned} \beta_1^{i,j} &= \dim\Big(\mathrm{Ker}(\partial_1^i)\Big) - \dim\Big(\mathrm{Im}(\partial_2^{i,j})\Big) \\ &= 1 - 1 \\ &= 0 \end{aligned} \tag{87}$$

as expected.

We now consider what the matrix representation of $\partial_2^{i,j}$ should be. We closely follow the steps outlined in Ref. [59], in particular the proof of Lemma 3.4 in that work.

The dimension 2 boundary matrix at scale j, projected onto simplices present at scale j is given by

$$\partial_2^j P_2^j = \begin{array}{c} \begin{matrix} ABC & ACD & ABD & BCD \end{matrix} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & -1 & -1 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \end{array} \begin{matrix} AB \\ BC \\ CD \\ AD \\ AC \\ BD \end{matrix}$$

As an aside, we note that the quantum algorithm in Ref. [18] defines the restricted boundary operator as $P_1^i \partial_2^j P_2^j$. We have been unable to verify that this quantum algorithm is able to correctly obtain the persistent Betti numbers. The matrix representation of this operator for our example is

$$P_1^i \partial_2^j P_2^j = \begin{array}{c} \begin{array}{cccc} ABC & ACD & ABD & BCD \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & -1 & -1 & 0 \end{pmatrix} \end{array} \begin{array}{c} AB \\ BC \\ CD \\ AD \end{array}$$

This operator has rank 3, and would therefore give $\beta_1^{i,j} = -2$, which is not possible. Another issue with this operator can be seen by considering its action on ABC (or any of the 2-simplices). The operator maps ABC to $AB + BC$, and we have that $ABC \in C_2(S^j)$, $AB, BC \in C_1(S^i)$, from which we would conclude that $ABC \in C_2^{i,j}(S^j)$. However, this is not the case, as $\partial_2^j(ABC) = BC - AC + AB$, and $AC \notin C_1(S^i)$. As discussed above, the chain group $C_2^{i,j}(S^j)$ is made up of a linear combination of the 2-simplices in S^j, and therefore we need a change of basis from simplices to chains, which is not present in the above expression for $\partial_2^{i,j}$. We provide additional discussion on issues encountered when using this expression for computing persistent Betti numbers at the end of this section.

To obtain the correct expression for the restricted boundary operator, we follow the procedure of Ref. [59], and consider the projection onto the 1-chains that are not in S_1^i

$$(I - P_1^i)\partial_2^j P_2^j = \begin{array}{c} \begin{array}{cccc} ABC & ACD & ABD & BCD \end{array} \\ \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \end{array} \begin{array}{c} AC \\ BD \end{array}$$

This operator maps 2-chains in j to 1-chains in j that are not in i. Therefore objects in its kernel are 2-chains in j that are in i. This coincides with the definition of $C_2^{i,j}(S^j)$, and therefore the elements of this group are in the kernel of $(I - P_1^i)\partial_2^j P_2^j := D_2^j$. We can identify these chains by performing column reduction on the matrix D_2^j. We need to find a non-singular matrix Y such that $R_2^j = D_2^j Y$ is column-reduced. Such a matrix Y can be found by doing a singular value decomposition of D_2^j. In this example, we use the matrix

$$Y = \begin{array}{c} \begin{array}{cccc} \alpha & \beta & \gamma & \delta \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \begin{array}{c} ABC \\ ACD \\ ABD \\ BCD \end{array}$$

such that

$$R_2^j = \begin{array}{c} \begin{array}{cccc} \alpha & \beta & \gamma & \delta \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \begin{array}{c} AC \\ BD \end{array}$$

We can see from R_2^j and Y that the columns indexed by α, δ are in the kernel of D_2^j, and correspond to the chains $\alpha = c_1 = ABC + ACD$, $\delta = c_2 = ABD + BCD$, as found earlier. As a result, Y has acted as a change of basis, from the simplicial basis, to a basis of chains that has, as a subset, a basis of $C_2^{i,j}(S^j)$.

The matrix representation for $\partial_2^{i,j}$ is then obtained by changing the column space of ∂_2^j using Y, and projecting into the columns corresponding to elements of $C_2^{i,j}(S^j)$, and the rows corresponding to 1-simplices in S^i:

$$\partial_2^{i,j} = \Big((\partial_2^j P_2^j) \cdot Y\Big)[AB : AD][\alpha, \delta]$$

$$
= \begin{array}{c} \\ AB \\ BC \\ CD \\ AD \\ AC \\ BD \end{array}
\begin{pmatrix}
\begin{array}{c} ABC \\ 1 \\ 1 \\ 0 \\ 0 \\ -1 \\ 0 \end{array} &
\begin{array}{c} ACD \\ 0 \\ 0 \\ 1 \\ -1 \\ 1 \\ 0 \end{array} &
\begin{array}{c} ABD \\ 1 \\ 0 \\ 1 \\ -1 \\ 0 \\ 1 \end{array} &
\begin{array}{c} BCD \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ -1 \end{array}
\end{pmatrix}
\cdot
\begin{pmatrix}
\begin{array}{cccc} \alpha & \beta & \gamma & \delta \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array}
\end{pmatrix}
\begin{array}{c} ABC \\ ACD \\ ABD \\ BCD \end{array}
[AB : AD][\alpha, \delta]
$$

$$
= \begin{array}{c} \\ AB \\ BC \\ CD \\ AD \\ AC \\ BD \end{array}
\begin{pmatrix}
\begin{array}{cccc} \alpha & \beta & \gamma & \delta \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ -1 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}
\end{pmatrix}
[AB : AD][\alpha, \delta]
$$

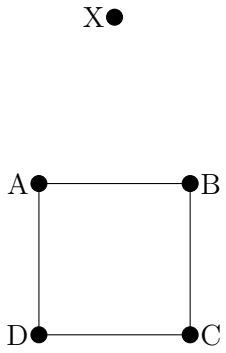
$$
= \begin{array}{c} \\ AB \\ BC \\ CD \\ AD \end{array}
\begin{pmatrix}
\begin{array}{cc} \alpha & \delta \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ -1 & -1 \end{array}
\end{pmatrix}
$$

This matrix maps 2-chains in $C_2(S^j)$ into 1-chains in $C_1(S^i)$, and has a rank of one. As a result, it correctly gives $\beta_1^{i,j} = 0$ for this filtration.

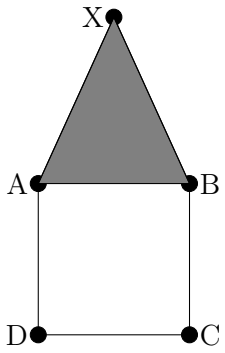
We see that finding the matrix representation of $\partial_{k+1}^{i,j}$ is more complex than finding the matrices $\partial_k^i, \partial_k^j$, etc. This results from the need to identify the subspace $C_{k+1}^{i,j}(S^j)$, and change to a basis that spans this subspace. In this example, this was achieved using the column reduction of D_2^j, and finding the rotation matrix Y. A more algorithmic approach was also introduced in Ref. [59], which builds the matrix representation of the persistent combinatorial Laplacian by expressing the matrix representation of $\partial_{k+1}^{i,j} \circ \left(\partial_{k+1}^{i,j}\right)^\dagger$ in terms of the Schur complement of the submatrix $\partial_{k+1}^j \left(\partial_{k+1}^j\right)^\dagger [\{I_k^j\}][\{I_k^j\}]$ in the matrix $\partial_{k+1}^j \left(\partial_{k+1}^j\right)^\dagger$, where the $\{I_k^j\}$ is the set of k-simplices in S^j that are not in S^i. The implementation of this Schur complement requires taking four submatrices of $\partial_{k+1}^j \left(\partial_{k+1}^j\right)^\dagger$, multiplying two of the submatrices by the Moore-Penrose pseudo-inverse of the third, and subtracting this product from the first submatrix. This approach was adapted for use in the quantum algorithm for finding persistent Betti numbers in Ref. [17]. Implementing the Schur complement requires quantum subroutines for block encoding the matrix $\partial_{k+1}^j \left(\partial_{k+1}^j\right)^\dagger$, performing projections into the subspaces S^i, S^j (and their complements), and being able to take the Moore-Penrose pseudo-inverse of block encoded matrices.

We now provide an additional discussion of issues encountered when using $P_k^i \partial_{k+1} P_{k+1}^j$ as a representation of the restricted boundary operator $\partial_{k+1}^{i,j}$, as was done in Ref. [18]. We consider the following pair of complexes:

$$S_0^i = \{A, B, C, D, X\}$$
$$S_1^i = \{AB, BC, CD, AD\}$$
$$S_2^i = \{\}$$



$$S_0^j = \{A, B, C, D, X\}$$
$$S_1^j = \{AB, BC, CD, AD, AX, BX\}$$
$$S_2^j = \{ABX\}$$



The persistent Betti number $\beta_1^{i,j}$ is 1 for this system. However, to compute the matrix representation of $P_1^i \partial_2 P_2^j$ we note that

$$\partial_2 P_2^j = \begin{pmatrix} ABX \\ 1 \\ -1 \\ 1 \end{pmatrix} \begin{matrix} \\ AB \\ AX \\ BX \end{matrix}$$

and

$$P_1^i \partial_2 P_2^j = \begin{pmatrix} ABX \\ 1 \\ 0 \\ 0 \end{pmatrix} \begin{matrix} \\ AB \\ AX \\ BX \end{matrix}$$

This matrix has a rank of 1. Hence, if we try to compute the persistent Betti number as

$$\begin{aligned} \beta_1^{i,j} &= \dim\Big(\mathrm{Ker}(\partial_1^i)\Big) - \dim\Big(\mathrm{Im}(\partial_2^{i,j})\Big) \\ &= 1 - 1 \\ &= 0 \end{aligned} \tag{88}$$

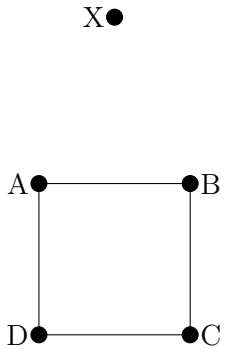
which is not the expected value.

E.2 Persistent Betti numbers from the quantum Zeno effect

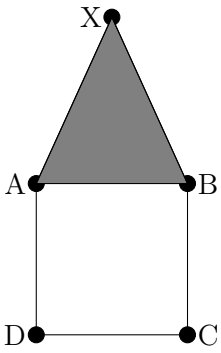
In Ref. [14] it is suggested that it is possible to compute the persistent Betti numbers using only the original quantum algorithm presented in Ref. [13] for computing Betti numbers. The suggested approach is to initially perform phase estimation with the combinatorial Laplacian at scale μ_1, repeating until an eigenvalue of 0 is found. Phase estimation is then performed again with the combinatorial Laplacian at scale $\mu_1 + \delta\mu$, and the hole is considered to persist for at least $\delta\mu$ if an eigenvalue of 0 is measured. Eventually, at scale μ_2 the hole will close fully, at which point the measured eigenvector no longer has eigenvalue 0. It is suggested that if the steps of phase estimation are performed at small enough increments of $\delta\mu$, the probability of projecting into the hole eigenstate stays high, making use of the quantum Zeno effect.

We have been unable to recover the suggested result, based on the following example. Consider the system

$$S_0^i = \{A, B, C, D, X\}$$
$$S_1^i = \{AB, BC, CD, AD\}$$
$$S_2^i = \{\}$$



$$S_0^j = \{A, B, C, D, X\}$$
$$S_1^j = \{AB, BC, CD, AD, AX, BX\}$$
$$S_2^j = \{ABX\}$$



The relevant distances in this diagram are: $d_{AB} = 2, d_{AX} \approx 2.42, d_{AC} = 2\sqrt{2}$. At scale i with $\mu_i = d_{AB}$, the hole $AB + BC + CD - AD$ is created. As the value of μ used to construct the combinatorial Laplacian is increased, there is no change until $\mu_j = d_{AX}$. At scale j the hole still persists. However, objects in the kernel of the combinatorial Laplacian are the harmonic representative of the homology group, and so can have unexpected forms [70]. In this case, the (unnormalized) zero eigenvector of Δ_1^j is given by $3(AB + BC + CD - AD) - \partial_2(ABX)$. The squared overlap between the (normalized) new and original holes is not equal to one. As a result, there is a non-zero probability of projecting into a different eigenstate, that is not in the kernel of Δ_1^j. Imagine a complex that consists of many copies of this system, separated by a large distance. As the quantum algorithm for Betti numbers works by sampling eigenstates at random, we are only able to count the fraction of eigenstates of Δ_1^i with zero eigenvalue. If we then try to project these zero eigenstates onto the eigenstates of Δ_1^j, we would erroneously conclude that some non-zero fraction of the holes had closed – leading us to believe that we have two types of holes, short-lived holes that close by stage j, and longer-lived holes that are still open at stage j. It appears that the Zeno-like procedure cannot conclusively determine persistent Betti numbers. The issue appears to be that while the parameter μ changes smoothly, the Laplacian changes discontinuously whenever a new simplex enters the filtration. As a result, the eigenstates undergo sudden jumps to include new basis states, required by the harmonic form. If these basis states were not present in the original eigenstate, then to ensure that probability is conserved, the state must also gain overlap with other (non-zero eigenvalued) eigenstates that also include the new basis states.

A possible modification to the idea outlined above would be to consider adiabatic state preparation between the initial combinatorial Laplacian Δ_k^i and the final combinatorial Laplacian Δ_k^j. For example, we could start in the ground state of Δ_k^i (whose normalized form is $0.5(AB + BC + CD - AD)$) and then time evolve the state under a time-dependent Hamiltonian $H(s) = (1-s)\Delta_k^i + s\Delta_k^j$, interpolating slowly between $s = 0$ and $s = 1$. We would then perform phase estimation with Δ_k^j. If we measured an eigenvalue of 0, we would conclude that the hole is still open. However, if we measured a non-zero eigenvalue we would conclude that the hole has closed. Unfortunately, we have been unable to obtain the correct result with this approach. The initial Laplacian has three relevant degenerate ground states; $0.5(AB + BC + CD - AD)$, AX, and BX. Only the first of these is present in the complex at stage i, and it is this state that we start in. As soon as $s \neq 0$, the degeneracy between the states is broken, and all three states have non-zero eigenvalues (until $s = 1$, when one of the states recovers an eigenvalue of 0). To see this rigorously, we expand the initial state in the non-zero eigenstates of $Q_0(\partial H/\partial s)Q_0 = Q_0(H(1) - H(0))Q_0$, where Q_0 is the projector onto the 3 relevant zero-valued eigenvectors of $H(0)$. This determines the amplitudes of the final states, assuming we move along the path infinitely slowly. We observe that the initial state has squared overlap of 0.945 and 0.055 with two of these eigenstates (it is also possible to check the squared overlap with the low-lying eigenstates

of $H(s \ll 1)$, and our results are in agreement). As a result, there will be a non-zero probability of finding the state in a non-zero eigenstate at the end of the adiabatic path. As with the example above, this would lead us to conclude that some fraction of the holes present in a complex have closed.