The Quantum Complexity of Computing Schatten p-norms

Chris Cade
School of Mathematics, University of Bristol, UK
chris.cade@bristol.ac.uk

Ashley Montanaro
School of Mathematics, University of Bristol, UK
ashley.montanaro@bristol.ac.uk

Abstract

We consider the quantum complexity of computing Schatten p-norms and related quantities, and find that the problem of estimating these quantities is closely related to the one clean qubit model of computation. We show that the problem of approximating $\mathrm{Tr}(|A|^p)$ for a log-local n-qubit Hamiltonian A and $p = \mathrm{poly}(n)$, up to a suitable level of accuracy, is contained in DQC1; and that approximating this quantity up to a somewhat higher level of accuracy is DQC1-hard. In some cases the level of accuracy achieved by the quantum algorithm is substantially better than a natural classical algorithm for the problem. The same problem can be solved for arbitrary sparse matrices in BQP. One application of the algorithm is the approximate computation of the energy of a graph.

2012 ACM Subject Classification Theory of computation → Quantum complexity theory, Theory of computation → Complexity classes

Keywords and phrases Schatten p-norm, quantum complexity theory, complexity theory, one clean qubit model

Digital Object Identifier 10.4230/LIPIcs.TQC.2018.4

Related Version A full version of the paper is available at `https://arxiv.org/abs/1706.09279`.

Funding CC was supported by the EPSRC. AM was supported by an EPSRC Early Career Fellowship (EP/L021005/1). No new data were created during this study.

1 Introduction

It is widely believed that quantum computers will be capable of solving certain computational problems more efficiently than any classical computer. However, the exact characterisation of the class of problems that allow for a quantum speedup is the subject of ongoing research. In complexity theory, this class is known as BQP [27] – the set of languages efficiently decidable by a uniform family of polynomial-size quantum circuits with bounded error. A useful way to understand and identify the types of problems that are efficiently solvable by a quantum computer, but unlikely to be efficiently solvable by a classical computer, is to find problems that are complete[1] for BQP; that is, problems that can be solved by a polynomial-time

[1] We note that what we are really referring to here are PromiseBQP-complete problems, since there are in fact no known BQP-complete problems. For a detailed discussion on this point see [14, 9].

13th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2018).
Editor: Stacey Jeffery; Article No. 4; pp. 4:1–4:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

quantum computer, and that any other problem in BQP can be reduced to. Intuitively, these are the very hardest problems in BQP.

Several BQP-complete problems are known, including approximating the Jones polynomial [1], estimating quadratically signed weight enumerators (QSWEs)[18], and estimating diagonal entries of powers of sparse matrices [14]. The latter problem is particularly interesting, since it is a relatively natural problem that is not obviously 'quantum' in nature.

Knill and Laflamme [18] showed that a more constrained version of the QSWE problem is efficiently solvable in the one clean qubit model of computation – an apparently non-universal model of quantum computation that is weaker than full quantum computation, but that can seemingly solve some problems more efficiently than a classical computer [25]. Understanding the power of such intermediate classes of computation could shed light on the types of problems that are efficiently solvable by a fully universal quantum computer.

We consider the computational complexity of estimating Schatten p-norms of matrices. We find that for certain values of p and certain families of matrices, this problem is closely related to the one clean qubit model of computation: depending on the accuracy of the estimation, the problem can be efficiently solved in the one clean qubit model, or is hard for this model of computation. We also consider similar quantities related to the spectra of matrices, such as the so-called "energy" of graphs [19, 10], and provide quantum algorithms for estimating them that are more efficient than any known classical algorithms.

1.1 The One Clean Qubit Model of Computation

The one clean qubit model of quantum computation initially arose as an idealised model for computation on highly mixed initial states, such as those that appear in NMR implementations [17]. In this model, we are given a quantum state consisting of a single 'clean' qubit in the pure state $|0\rangle$, and n qubits in the maximally mixed state. This can be represented by the density matrix

$$\rho = |0\rangle\langle 0| \otimes \frac{I_{2^n}}{2^n}.$$

We then apply an arbitrary polynomial-sized quantum circuit to ρ, and measure the first qubit in the computational basis. Following [17], we will refer to the class of problems that can be solved in polynomial time using this model of computation as DQC1 – deterministic quantum computation with a single clean qubit.

The canonical problem that can be solved in this model is that of estimating the normalised trace of a $2^n \times 2^n$ unitary matrix U corresponding to a polynomial-size quantum circuit. This is achieved by applying a controlled version of U to ρ, where the clean qubit is used as the control qubit and is put into the state $(|0\rangle + |1\rangle)/\sqrt{2}$ using a Hadamard gate. More precisely, we apply the controlled-U operator to the state

$$\rho' = \frac{1}{2}(|0\rangle + |1\rangle)(\langle 0| + \langle 1|) \otimes \frac{I_{2^n}}{2^n}$$

and then apply a Hadamard gate to the first qubit, before measuring it. The probability of measuring zero is $\frac{1}{2} + \frac{1}{2}\frac{\mathrm{Re}(\mathrm{Tr}(U))}{2^n}$, which can be estimated up to accuracy ϵ by repeating the procedure $O(1/\epsilon^2)$ times. The imaginary part of the trace of U can be estimated similarly by starting with the first qubit in the state $\frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle)$. This problem has been shown to be complete for the class DQC1 [26].

More generally, we might consider the DQCk model of computation. That is, deterministic quantum computation with k pure qubits. If $k = O(\log(n))$, then the DQCk model is equivalent to DQC1 [26]. This result is important for us since the quantum circuit that we apply to the initial state may require a number of ancilla qubits initialised to $|0\rangle$ in order to correctly perform its computation. For example, if the quantum circuit implementing the unitary U performs the phase estimation routine, then it will usually require an additional $O(\log n)$ clean qubits. In the context of estimating the trace of a unitary matrix, this result tells us that it is possible in DQC1 to compute the trace of a sub-matrix whose size is an inverse-polynomially large fraction of the size of the input matrix.

1.1.1 DQC1-complete Problems

Knill and Laflamme [17] showed that the problem of estimating a coefficient in the Pauli decomposition of a quantum circuit, up to polynomial accuracy, is complete for the class DQC1. In fact, the aforementioned problem of estimating the normalised trace of a quantum circuit is a special case of this problem [26]. Shor and Jordan [26] added to the relatively short list of DQC1-complete problems by showing that the problem of estimating the 'trace closure' of Jones polynomials is also complete for the class DQC1. Finally, Brandão [6] showed that two problems related to Hamiltonians were DQC1-complete: computing the partition function of a class of (quantum) Hamiltonians, and computing the sum of all eigenvalues of a Hamiltonian that fall between two given energy levels.

These quantities appear to be hard to compute classically, and therefore the one clean qubit model of computation seems to be more powerful than classical computation. However, it is unlikely that DQC1 contains all of BQP [25], and thus this model of computation appears to have a computational power that is somewhere in between BPP and BQP. Some evidence in this direction was recently provided by Morimae [21], who built on earlier work ([22]) to show that the output distribution of the one clean qubit model is difficult to sample from classically up to constant total variation distance error, provided that some complexity theoretic conjectures hold.

Here we show that the problem of computing Schatten p-norms of matrices is also closely related to the class DQC1.

1.2 Schatten p-norms and Graph Energy

Schatten p-norms are ubiquitous in Quantum Information theory (see for example [24, 3, 12]). This family of matrix norms includes the three most commonly used norms in quantum information theory: the Schatten 1-norm is more commonly called the trace norm, the Schatten 2-norm is also known as the Frobenius norm, and the Schatten ∞-norm is called the operator norm or spectral norm. Here we consider the normalised Schatten p-norm, defined as

$$\|A\|_p := \left(\frac{\sum_j |\lambda_j|^p}{2^n}\right)^{1/p}$$

for a $2^n \times 2^n$ Hermitian matrix A, where the sum ranges over the eigenvalues of A.

For instance, the Schatten 1-norm is the average of the absolute values of the eigenvalues of A,

$$\|A\|_1 = \frac{\mathrm{Tr}(|A|)}{2^n} = \frac{\sum_j |\lambda_j|}{2^n}.$$

If we consider the matrix A to be the adjacency matrix of a graph, this quantity is known as the 'Graph Energy', and has applications in chemistry, where it is related to the total electron energy of a class of organic molecules [19, 10]. More generally, quantities relating to the spectra of adjacency matrices are used throughout Graph Theory to reveal information about the graphs that they represent. In the present work, we consider some 'global' properties of the spectra of matrices and graphs – i.e. those of the form $\mathrm{Tr}(f(A))/2^n$, for some suitably chosen function f. The Schatten p-norms are examples of such quantities.

1.3 Our results

We study the complexity of approximately computing Schatten p-norms of sparse matrices and relate this to quantum computation. We consider Hermitian matrices of size $2^n \times 2^n$, where at most $d = \mathrm{poly}(n)$ entries in each row are non-zero, and call such matrices d-sparse. One fairly natural class of sparse matrices that can be expressed concretely is the class of 'log-local' Hamiltonians. That is, k-local n-qubit Hamiltonians, with $k = O(\log n)$ - i.e. Hermitian matrices that can be written as a sum $A = \sum_{j=1}^{m} A_j$, for some m, where each A_j is a Hermitian matrix that acts non-trivially on at most $k = O(\log n)$ qubits. We assume that we are given the individual matrices A_j directly, that $\|A_j\| = O(\mathrm{poly}(n))$ for all j, and that $m = \mathrm{poly}(n)$. Throughout this work, we use $\|A\|$ to denote the operator norm of A.

▶ **Theorem 1.** *Let A be a sparse Hermitian matrix on n qubits, and let $p, 1/\epsilon = O(\mathrm{poly}(n))$. Then the problem of estimating $\frac{\mathrm{Tr}(|A|^p)}{2^n}$ up to additive accuracy $\epsilon\|A\|^p$ is contained in BQP. If the matrix A is* log*-local, then this problem is also contained in DQC1.*

▶ **Theorem 2.** *Let A be a* log*-local Hermitian matrix on n qubits. Then the problem of estimating $\frac{\mathrm{Tr}(|A|^p)}{2^n}$ up to additive accuracy $\epsilon\left(\frac{\|A\|}{2}\right)^p$ for arbitrary $p, 1/\epsilon = O(\mathrm{poly}(n))$ is hard for the class DQC1.*

The BQP case of Theorem 1 follows from the result of Janzing and Wocjan [13], who gave a BQP algorithm for estimating diagonal entries of $f(A)$, for a sparse matrix A and an appropriate function f which can be taken to be $f(x) = |x|^p$.

We therefore see that the problem of computing Schatten p-norms for $p = O(\mathrm{poly}(n))$ is closely related to the one clean qubit model of computation. By contrast, for different values of p the problem is related to other classes of computation. For instance, $\|A\|_\infty$ is the operator norm of A, and the problem of computing it approximately is QMA-complete[2], even for 2-local Hamiltonians. To see this, suppose we have some upper bound $\Delta = O(\mathrm{poly}(n))$ on the largest eigenvalue of a 2-local n-qubit Hamiltonian A. Define the matrix $B := \Delta I_{2^n} - A$. Then the largest eigenvalue of B (in absolute value) corresponds to the smallest eigenvalue of A. Hence, if we can compute the smallest eigenvalue of A, then we can compute $\|B\|$, and vice versa. Since the problem of estimating the smallest eigenvalue of a k-local Hamiltonian is QMA-complete for $k \geq 2$ [15], this implies QMA-completeness of the problem of estimating the operator norm of a 2-local Hamiltonian.

Note that the required accuracies of the estimates in Theorems 1 and 2 differ by a factor of $1/2^p$. Unfortunately, we were unable to reconcile this difference, and therefore we did not find a variant of the problem that is *complete* for DQC1.

Theorem 1 gives us the following corollary:

▶ **Corollary 3.** *Let A be a* log*-local matrix corresponding to the adjacency matrix of a 2^n-vertex graph G, and let $p, 1/\epsilon = O(\mathrm{poly}(n))$. The normalised Graph Energy of G, $\mathrm{Tr}(|A|)/2^n$, can be estimated up to additive accuracy $\epsilon\|A\|$ in DQC1.*

[2] For a definition of the class QMA, see [27].

In proving Theorem 1, we also show that there exists a polynomial-time quantum algorithm (in DQC1) for estimating $\mathrm{Tr}(A^p)/2^n$ up to error $\epsilon\|A\|^p$ for $1/\epsilon, p \in O(\mathrm{poly}(n))$. This is useful in the context of graph theory because it allows for an estimation of the expected number of closed walks that start from each vertex in a 2^n-vertex graph. To obtain these algorithms, we prove a more general result:

▶ **Lemma 4.** *For a log-local Hamiltonian A, and any* log-*space polynomial-time computable function $f : I \to [-1, 1]$ (where I contains the spectrum of A) that is Lipschitz continuous with constant K (i.e. $|f(x)-f(y)| \leq K|x-y|$ for all $x, y \in I$), there exists a DQC1 algorithm to estimate $\mathrm{Tr}(f(A))/2^n = \sum_j f(\lambda_j)/2^n$ up to additive accuracy $\epsilon(K + 1)$, where λ_j denote the eigenvalues of A, and $\epsilon = \Omega(1/\mathrm{poly}(n))$.*

Often, one is interested in calculating the properties of general sparse matrices. We note that it is easy to give a quantum algorithm for estimating the above quantities for sparse matrices by making use of a result of Janzing and Wocjan [14, 13], who give a BQP algorithm for estimating the diagonal entries of $f(A)$, for some function f that satisfies certain continuity constraints, but this comes at the expense of moving to the class BQP.

It is interesting to note that the results from [6] also make use of log-local Hamiltonians. In both these and our results, it is not clear how to drop the restriction of log-locality without losing the fact that the various problems are contained in DQC1.

1.3.1 Estimating $\|A\|_p$

Given a log-local n-qubit Hamiltonian A, the algorithm of section 3 outputs

$$\mathrm{Tr}(|A|^p)/2^n \pm \epsilon\|A\|^p.$$

By taking the pth root, we obtain an estimate of $\|A\|_p$ of the form

$$\left(\frac{\mathrm{Tr}(|A|^p)}{2^n} \pm \epsilon\|A\|^p\right)^{1/p} = \|A\|_p \left(1 + \frac{2^n\epsilon\|A\|^p}{\mathrm{Tr}(|A|^p)}\right)^{1/p}.$$

The error will be small when $\mathrm{Tr}(|A|^p)$ takes a value close to its maximum of $2^n\|A\|^p$. In the best case, the relative error is close to $(1 + \epsilon)^{1/p}$. This suggests that in these 'good' cases, our algorithm can estimate $\|A\|_p$ up to a reasonable additive error in polynomial time. On the other hand, we can always bound

$$\frac{2^n\|A\|^p}{\mathrm{Tr}(|A|^p)} \leq \frac{2^n\|A\|^p}{2^n|\lambda_{\min}|^p} = \kappa(A)^p,$$

where $\lambda_{\min}$ is the minimal eigenvalue of A in absolute value, and $\kappa(A) = \|A\|\|A^{-1}\|$ is the condition number of A. In this case the relative error is at most $(1 + \epsilon\kappa(A)^p)^{1/p}$.

Since we consider $p = \mathrm{poly}(n)$, the algorithm allows us to achieve relative error close to $\kappa(A)$ by taking $\epsilon = 1 - 1/\kappa(A)^p \approx 1$. Alternatively, we could achieve relative error $(1 + \delta)$ for some $\delta = O(1/\mathrm{poly}(n))$ by setting $\epsilon = ((1 + \delta)^p - 1)/\kappa(A)^p$. In this case, we sacrifice the run-time of the algorithm in order to improve the accuracy.

1.4 Relation to Previous Work

Our techniques are similar to those used in [14] and [11]. In particular, we use the same combination of Hamiltonian simulation and phase estimation for estimating and manipulating the eigenvalues of a Hermitian matrix. To show DQC1-hardness, we use techniques from the Hamiltonian complexity literature, and in particular ideas due to Kitaev et al. [16, 15].

By using a previous result of Janzing and Wocjan [14], we can obtain a BQP algorithm for estimating $\mathrm{Tr}(A^p)/2^n$ for general sparse matrices; however, it is not clear how to implement this algorithm in DQC1, since it uses $O(n)$ ancilla qubits for the Hamiltonian simulation step. In [14], the authors describe a polynomial-time quantum algorithm for estimating the diagonal entries of the matrix A^p up to error $\epsilon\|A\|^p$, for $\epsilon = O(1/\operatorname{poly}(n))$, and show that this problem is in fact BQP-complete for sparse symmetric matrices. The problem remains BQP-complete even for matrices with only $0, \pm 1$ entries.

1.5 Comparison with Classical Algorithms

We were not able to find any previous results in the literature regarding the complexity of estimating the above quantities for sparse matrices. In Appendix B, we give a classical algorithm for estimating the normalised trace of a sparse matrix raised to some power, and prove some bounds on the accuracy that this algorithm can achieve.

We find that for some types of matrix, the value $\mathrm{Tr}(A^p)/2^n$ can be estimated efficiently classically, and for others, a quantum algorithm appears to have some advantage over a classical one. In Appendix B, we prove the following:

▶ **Theorem 5.** *Given a $2^n \times 2^n$, d-sparse matrix A, there exists a classical algorithm to estimate $\mathrm{Tr}(A^p)/2^n$ up to accuracy $\epsilon d^p \|A\|_{\max}^p$ in time that is polynomial in n, p and $1/\epsilon$, where $\epsilon = O(1/\operatorname{poly}(n))$ and $\|A\|_{\max}$ is used to denote the maximum absolute size of an entry in A.*

Therefore, in the cases where $\|A\| \ll d\|A\|_{\max}$, we can get an advantage by making use of the algorithm of Theorem 1. We find that for certain classes of random graph (namely power-law graphs), the BQP algorithm for computing $\mathrm{Tr}(A^p)/2^n$ obtains a quadratic improvement in accuracy over the corresponding classical algorithm.

For log-local Hamiltonians and constant p, there exists an efficient *exact* classical algorithm for computing $\mathrm{Tr}(A^p)$. By using conventional matrix multiplication, it is possible to calculate the value of A^p by multiplying the individual matrices A_j. This can be seen from the expression for $\mathrm{Tr}(A^p)$:

$$\mathrm{Tr}(A^p) = \sum_{j_1, j_2, \ldots, j_p} \mathrm{Tr}(H_{j_1} H_{j_2} \cdots H_{j_p}),$$

where each index j_i ranges from 1 to m. Every H_{j_i} is k-local, and the complexity of multiplying a k-local matrix by an l-local matrix is $O(2^{3(k+l)})$ (using a naive algorithm), and results in a $(k+l)$-local matrix. If we perform the matrix multiplications from left to right, then, for each term in the sum, the first multiplication will take time $O(2^{3(2k)})$, the second $O(2^{3(3k)})$, and so on, until the final multiplication takes time $O(2^{3(pk)})$. There will be $p-1$ of these multiplications performed in total, with each taking at most $O(2^{3 \cdot pk})$ time, and hence the trace of $A_{j_1} A_{j_2} \cdots A_{j_p}$ can be calculated in $O(2^{3 \cdot pk})$ steps. There are m^p terms in the sum, and therefore the complexity of the entire computation is $O(m^p 2^{3 \cdot pk})$.

If we take $k = O(\log n)$ (i.e. take A to be a log-local Hamiltonian), the time complexity is $m^p n^{O(p)}$. For $p = O(1)$, this time complexity is polynomial and the output of this algorithm is better than the corresponding quantum algorithm, as it computes the desired value exactly.

Note that the problem of computing $\mathrm{Tr}(|A|^p)$ appears to be substantially harder classically for odd p, since it cannot be found by simply computing powers of a matrix, and instead requires more knowledge about the eigenvalues of A.

1.6 Organisation

We begin by providing a proof of Theorem 2 in Section 2. Section 3 describes a proof of Theorem 1, via an algorithm in the one clean qubit model that can estimate $\mathrm{Tr}(f(A))/2^n$ for a $2^n \times 2^n$ log-local matrix A and an appropriately continuous function f. Following this, in Section 4, we compare the performance of the quantum algorithm with its classical counterpart, which is described in Appendix B. Finally, appendix A contains a detailed analysis of the quantum algorithm used in Section 3.

2 Estimating $\mathrm{Tr}(|A|^p)/2^n$ is DQC1-hard

In this section, we show that the problem of estimating $\mathrm{Tr}(|A|^p)/2^n$ for a $2^n \times 2^n$ log-local Hamiltonian A up to a given accuracy is hard for the class DQC1. More precisely, we assume that we have access to an algorithm that can estimate $\mathrm{Tr}(|A|^p)/2^n$ up to accuracy $\epsilon\left(\frac{\|A\|}{2}\right)^p$, for $\epsilon = O(1/\mathrm{poly}(n))$ and $p = \mathrm{poly}(n)$, and show that this implies that we can solve any problem contained in DQC1.

To do this we show that, given as input a real unitary U (implemented by some polynomial-sized quantum circuit acting on n qubits), it is possible to construct a log-local Hamiltonian A such that $\mathrm{Tr}(|A|^p)/2^n = \mathrm{Tr}(U)/2^n$, for some $p = \mathrm{poly}(n)$. Furthermore, we show that an estimation accuracy of $\epsilon\left(\frac{\|A\|}{2}\right)^p$ is sufficient to provide an estimate of $\mathrm{Tr}(U)/2^n$ up to accuracy $1/\mathrm{poly}(n)$. This problem is complete for the class DQC1 [26], which implies that the problem of estimating $\mathrm{Tr}(|A|^p)/2^n$ up to the stated accuracy is DQC1-hard.

The construction is based on ideas from Hamiltonian complexity, and in particular Kitaev's clock construction for the local Hamiltonian problem [2]. We assume that we have a decomposition $U = U_{M-1}...U_1U_0$ of the circuit into M elementary gates. Since U is described by a polynomial-sized circuit, we have $M = \mathrm{poly}(n)$. We add $\lceil \log M \rceil$ additional qubits to act as a 'clock' register, which is used to control the application of the individual unitaries, and define a unitary operator

$$W := \sum_{l=0}^{M-1} |l+1\rangle\langle l| \otimes U_l,$$

where addition is taken to be modulo M. It is straightforward to check that

$$W^M = \sum_{l=0}^{M-1} |l\rangle\langle l| \otimes U_{l+M}...U_{l+2}U_{l+1}.$$

Then we have

$$\mathrm{Tr}(W^M) = \sum_{l=0}^{M-1} \mathrm{Tr}(|l\rangle\langle l|) \cdot \mathrm{Tr}(U_{l+M}...U_{l+2}U_{l+1}) = \sum_{l=0}^{M-1} \mathrm{Tr}(U_M...U_2U_1) = M\,\mathrm{Tr}(U),$$

where the second step follows from invariance of the trace under cyclic permutations.

W is log-local with $m = \mathrm{poly}(n)$ terms, since each clock operator $|l+1\rangle\langle l|$ acts on $\lceil \log M \rceil$ qubits, and each of the unitaries U_l act on at most $O(1)$ qubits each. Define the Hermitian matrix

$$A := \frac{1}{2}(W + W^\dagger).$$

Then the trace of A^M gives the real part of the trace of $\frac{2W^M}{2^M}$, since A^M equals $1/2^M(W^M + {W^\dagger}^M)$ plus some other powers of W and $W^\dagger$ that are traceless (since the clock unitaries can only have a trace if they return the clock state back to its initial state, which takes at least M applications of W), and therefore do not contribute to the trace of A^M.

W is a $2^{n+\lceil \log M \rceil} \times 2^{n+\lceil \log M \rceil}$ unitary matrix, and so we have $\|A\| \leq 1$. Thus, given the ability to estimate the normalised trace of A^p up to accuracy $\left(\frac{\|A\|}{2}\right)^p \epsilon$, we can estimate the value of $\mathrm{Re}[\mathrm{Tr}(U)]/2^n$ up to accuracy $1/\operatorname{poly}(n)$, which is the level of accuracy required for the class DQC1. To see this, we observe that, taking $p = M$ and assuming (without loss of generality) that M is a power of 2 (which also means that $|A|^M = A^M$),

$$\frac{\mathrm{Tr}(A^M)}{2^{n+\log M}} \pm \frac{\epsilon}{2^M} = \frac{2\,\mathrm{Re}(\mathrm{Tr}(W^M))}{2^M 2^{n+\log M}} \pm \frac{\epsilon}{2^M} = \frac{2M\,\mathrm{Re}(\mathrm{Tr}(U))}{M 2^M 2^n} \pm \frac{\epsilon}{2^M}.$$

Multiplying by 2^M, we obtain $\frac{\mathrm{Re}(\mathrm{Tr}(U))}{2^n} \pm \epsilon$, which is precisely the quantity that is DQC1-hard to compute. This is sufficient to show that the problem of estimating $\mathrm{Tr}(A^p)/2^n$ up to accuracy $\left(\frac{\|A\|}{2}\right)^p \epsilon$ for a log-local n-qubit Hamiltonian is hard for the class DQC1.

Note that we were not able to use standard techniques from the Hamiltonian complexity literature to make this construction work for k-local Hamiltonians with constant k [15, 16]. These techniques involve the introduction of a larger clock space that is then acted upon by k-local Hamiltonians. A term is then added to the Hamiltonian to 'penalise' invalid clock states and prevent them from contributing to the ground state energy. In our case, we care about the entire space on which the Hamiltonian acts and not just the subspace containing the valid clock states, and therefore the invalid clock states contribute to the trace of A^M in a non-trivial way.

3 Estimating $\mathrm{Tr}(|A|^p)/2^n$ is in DQC1

Here we show that the problem of estimating $\mathrm{Tr}(|A|^p)/2^n$ for a log-local Hamiltonian A, up to reasonable error, is in DQC1. In precise terms, we are given a k-local n-qubit Hamiltonian A, with $k = O(\log n)$; then the problem is to estimate $\mathrm{Tr}(|A|^p)/2^n$ up to error $\epsilon\|A\|^p$, for some integer $p = O(\operatorname{poly}(n))$ and accuracy $\epsilon = \Omega(1/\operatorname{poly}(n))$. Our approach is to construct a unitary U such that the normalised trace of U approximates the normalised trace of $|A|^p$. We show that this construction can be performed in polynomial time (that is, the unitary U takes $\operatorname{poly}(n, p, 1/\epsilon)$ time to implement). Using this approach, we can use the DQC1 model to compute the normalised trace of the matrix $|A|^p$, hence showing that this problem is contained in DQC1. We will use the following corollary of Lemma 4:

► **Corollary 6.** *For a log-local Hamiltonian A, and any* log*-space polynomial-time computable function $f : I \to \mathbb{R}$ (where I contains the spectrum of A) that is Lipschitz continuous with constant K' (i.e. $|f(x) - f(y)| \leq K'|x - y|$ for all $x, y \in I$), there exists a DQC1 algorithm to estimate $\mathrm{Tr}(f(A))/2^n = \sum_j f(\lambda_j)/2^n$ up to additive accuracy $\epsilon(K' + f_{\max})$, where λ_j denote the eigenvalues of A, $\epsilon = \Omega(1/\operatorname{poly}(n))$, and $f_{\max}$ is the supremum of $|f|$ on the interval I.*

The proof of Lemma 4 (and hence the above corollary) is split into roughly three parts. The first part, in Section 3.1, describes how the algorithm works, via a description of the unitary that is constructed from the input matrix. Following this, Section 3.2 discusses the accuracy and failure probability of the algorithm, and finally, Section 3.3 shows that the number of ancilla qubits required (and therefore the number of pure qubits needed) to implement the algorithm is at most $O(\log n)$.

3.1 Constructing the Unitary

We are given a log-local Hamiltonian A with eigenvectors $|\psi_j\rangle$ and corresponding eigenvalues λ_j. The basic idea is to construct a unitary U whose eigenvalues correspond to the eigenvalues of A in a useful way. In particular, we construct a polynomial-sized circuit whose associated unitary has eigenvalues λ'_j such that $\lambda'_j = f'(\lambda_j)$, for some function f' that depends on f.

The first step is to use Hamiltonian simulation to implement the unitary e^{iA}, which has eigenvalues $e^{i\lambda_j}$ for each eigenvector $|\psi_j\rangle$ of A. Section 3.4 discusses the time complexity of this part of the circuit. Then the circuit performs the following sequence of operations, which we will describe in terms of their effects on an eigenvector $|\psi_j\rangle$ of A and an arbitrary single qubit state of the form $\alpha|0\rangle + \beta|1\rangle$. We use $|0\dots0\rangle$ to denote an arbitrarily large ancilla register (with each qubit initialised to 0), and assume that both the phase estimation and Hamiltonian simulation parts of the circuit work perfectly.

1. Apply phase estimation on e^{iA} with the input $|\psi_j\rangle$, to obtain an estimate of the eigenvalue λ_j:

$$|\psi_j\rangle(\alpha|0\rangle + \beta|1\rangle)|0\dots0\rangle \mapsto |\psi_j\rangle(\alpha|0\rangle + \beta|1\rangle)|\lambda_j\rangle$$

2. Perform controlled phase rotations, where the phase depends on a function f of λ_j contained in the 3rd register (for example, $f(x) = x^p$):

$$|\psi_j\rangle(\alpha|0\rangle + \beta|1\rangle)|\lambda_j\rangle \mapsto |\psi_j\rangle(\alpha e^{i\arccos(f(\lambda_j))}|0\rangle + \beta e^{-i\arccos(f(\lambda_j))}|1\rangle)|\lambda_j\rangle$$

3. Undo the phase estimation to uncompute the value in the 3rd register:

$$\mapsto |\psi_j\rangle(\alpha e^{i\arccos(f(\lambda_j))}|0\rangle + \beta e^{-i\arccos(f(\lambda_j))}|1\rangle)|0\dots0\rangle$$

This gives us a unitary U that performs the mapping

$$|\psi_j\rangle(\alpha|0\rangle + \beta|1\rangle)|0\dots0\rangle \mapsto (\alpha e^{+i\arccos(f(\lambda_j))}|\psi_j\rangle|0\rangle + \beta e^{-i\arccos(f(\lambda_j))}|\psi_j\rangle|1\rangle)|0\dots0\rangle$$

for each eigenvector $|\psi_j\rangle$ of A. Therefore, for each eigenvalue λ_j of A, U has two corresponding eigenvalues $e^{\pm i\arccos(f(\lambda_j))}$.

By using the results described in Section 1.1, we can compute the trace of a sub-matrix of U in the one clean qubit model, provided that the number of ancilla qubits used is $O(\log n)$ (we check that this is indeed the case at the end of this section). In particular, we compute the trace of U', the sub-matrix of U obtained by fixing the ancilla qubits (except the one explicitly mentioned above) to $|0\rangle$. Then the trace of U' is

$$\begin{aligned}\mathrm{Tr}(U') &= \sum_j e^{\pm i\arccos(f(\lambda_j))} = \sum_j \cos(\pm\arccos(f(\lambda_j))) + i\sin(\pm\arccos(f(\lambda_j))) \\ &= \sum_j 2\cos(\arccos(f(\lambda_j))) + i\sin(\arccos(f(\lambda_j))) - i\sin(\arccos(f(\lambda_j))) \\ &= \sum_j 2f(\lambda_j).\end{aligned}$$

3.2 Error Analysis

Errors can arise in three places. Firstly, we will have some error in the Hamiltonian simulation part of the circuit. Secondly, there will be errors in estimating eigenvalues by using the phase estimation routine. And finally, there will be some error in the estimation of the normalised trace of U from using the one clean qubit model. In Appendix A, we consider the effect of all three sources of error, and show that we can estimate $\frac{1}{2^n}\sum_j f(\lambda_j)$ with additive error at most $\epsilon(K+1)$, for any $\epsilon = \Omega(1/\operatorname{poly}(n))$, where K is the Lipschitz constant of f. The analysis in this section is analogous to that of [14], since we use the same method for estimating an eigenvalue of A via simulation of e^{iA}, but uses different methods to bound the errors introduced by phase estimation and Hamiltonian simulation.

3.3 How many clean qubits are needed?

Here we consider how many clean qubits are required to implement the circuit described in Section 3.1 up to the desired accuracy. Any time the circuit uses ancilla qubits, these qubits will generally need to be initialised in the all-zeros state – that is, they must be under our control, and be 'clean'. As discussed in Section 1.1, we can use $O(\log n)$ clean qubits without changing the model of computation. In this section we argue that the implementation of the circuit described above requires no more than $O(\log n)$ ancilla qubits.

The two main parts of the circuit are the phase estimation routine, and Hamiltonian simulation. The rest of the circuit consists of more basic operations that require only a constant number of ancilla qubits (provided that the function f we choose is sufficiently easy to compute).

To achieve the accuracy stated in the previous section, we show in Appendix A that the phase estimation part of the circuit must be able to achieve an additive accuracy of $\frac{1}{2}\epsilon(K+1)$, for which it only requires $O(\log n)$ ancilla qubits. Hence, the number of clean qubits required to implement the phase estimation part of the circuit is $O(\log n)$.

In order to implement the simulation of the Hamiltonian A, we can use techniques based on the Lie-Trotter product formula [20]. This requires no more than a constant number of ancilla qubits, and, since we assume that we are given the Hamiltonian directly as a set of m individual Hamiltonians that each act on $O(\log n)$ qubits, there are no ancilla qubits required to 'load' the input into the system, which would be the case if we considered the case where the input Hamiltonian is specified by an oracle (it is precisely for this reason that we define the problem in terms of a log-local Hamiltonian rather than a sparse Hamiltonian). In our case, we can run a polynomial-time classical algorithm to compute the quantum circuit required to implement the unitary e^{iA}, given such a description of A. This is discussed more fully in the following section.

3.4 Simulating log-local Hamiltonians

We are required to implement the unitary e^{iA} for some log-local Hamiltonian A. We are limited to using at most $O(\log n)$ ancilla qubits, which rules out the more advanced Hamiltonian simulation techniques that are based on quantum walks (e.g. [5]). Instead, we use the 'vanilla' version of Hamiltonian simulation, which is based on the Lie-Trotter product formula [20].

We are given a log-local n-qubit Hamiltonian A, and wish to implement a unitary operator that approximates e^{iAt} for some value of t, up to a specified accuracy δ (in the operator norm). That is, we want to construct, in classical polynomial time, a quantum circuit that implements a unitary operator V such that

$$\|V - e^{iAt}\| \leq \delta.$$

It is straightforward to check that the standard techniques, which are usually presented for $O(1)$-local Hamiltonians, indeed work for log-local Hamiltonians, and allow us to simulate e^{iAt} up to accuracy δ in time

$$O(\operatorname{poly}(m, n, \tau, 1/\delta)),$$

where $\tau = t\|A\|$, using a circuit that can be computed by a polynomial-time classical algorithm[3]. The time complexity could be improved by the use of more complicated simulation techniques [4], but we do not consider this here.

In the circuit described in Section 3.1, we set $t = 1$, and require that $\delta = O(1/\operatorname{poly}(n))$. Thus, the time taken to implement the Hamiltonian simulation part of the circuit will be $O(\operatorname{poly}(n))$.

3.5 Proof that computing $\mathrm{Tr}(|A|^p)/2^n$ is in DQC1

The proof of Theorem 1, which states that the problem of estimating $\mathrm{Tr}(|A|^p)/2^n$ up to error $\epsilon\|A\|^p$ is in DQC1 for $p, 1/\epsilon = \operatorname{poly}(n)$, follows almost immediately from Lemma 4. The same proof also applies to the problem of estimating $\mathrm{Tr}(A^p)/2^n$. It is straightforward to check that, on the interval $[-b, b]$, both $f(x) = x^p$ and $f(x) = |x|^p$ are Lipschitz continuous with Lipschitz constant $K = pb^{p-1}$. Furthermore, $f_{\max} = b^p$ for both functions. In our case we can take $b = \|A\|$, since f is a function of the eigenvalues of A. Putting these values into Corollary 6, and replacing ϵ with $\frac{\epsilon}{p/\|A\|+1}$, we obtain an estimate of $\frac{\mathrm{Tr}(|A|^p)}{2^n}$ up to accuracy $\epsilon\|A\|^p$. Furthermore, this estimate can be obtained in DQC1 in time that is polynomial in n and inverse polynomial in ϵ.

4 Quantum vs. Classical

Here we compare the complexities of the (BQP) quantum and classical algorithms for computing $\mathrm{Tr}(A^p)$, for random $N \times N$ matrices. Recall that the quantum algorithm has an accuracy of $\epsilon\|A\|^p$, and that the classical algorithm has an accuracy of $\epsilon d^p \|A\|^p_{\max}$, where $p = \operatorname{polylog}(N)$.

In the event that $\|A\| \ll d$, the quantum algorithm achieves an improvement in accuracy over the classical algorithm. However, since the quantum algorithm requires the matrix A to be sparse, we must restrict our attention to only sparse matrices that have this property. Towards this end, we will begin by considering a general model for random graphs, and introduce some results that relate the degrees of the vertices of the graph to the eigenvalues of the adjacency matrix. Following this, we will consider how these results apply to sparse graphs.

4.1 Random Graphs

We consider a general model for unweighted random graphs (see e.g. [7]), in which each vertex v is associated with a weight w_v. Then a random graph G is constructed by assigning an edge independently to each pair of vertices (i, j) with probability $\frac{w_i w_j}{\sum_i w_i}$, such that the expected degree of vertex v is given by w_v. Denote by d the maximum expected degree, and by $\tilde{d}$ the value

$$\tilde{d} := \frac{\sum_{i=1}^N w_i^2}{\sum_{i=1}^N w_i}.$$

Then we have the following results from [7]:

[3] The details of this simulation can be found in the full version of this paper.

▶ **Theorem 7.** *If $\tilde{d} > \sqrt{d}\ln N$, then as $N \to \infty$ the largest eigenvalue of a random graph $G(w)$ is almost surely $(1+o(1))\tilde{d}$.*

▶ **Theorem 8.** *If $\sqrt{d} > \tilde{d}\ln^2 N$, then as $N \to \infty$ the largest eigenvalue of a random graph $G(w)$ is almost surely $(1+o(1))\sqrt{d}$*

Intuitively, $\|A\|$ is (asymptotically) the maximum of $\sqrt{d}$ and $\tilde{d}$ if the two values $\sqrt{d}$ and $\tilde{d}$ are far apart (i.e. by a power of $\log N$).

4.2 Restriction to Sparse Graphs

We are interested in sparse graphs – i.e. those in which the degree of every vertex is $O(\text{polylog}(N))$. If we use the random graph model above, and set $d = \Theta(\log^2 N)$, then if we allow all vertices to have an expected degree similar to d, then by Theorem 7, $\|A\| = (1+o(1))d$ almost surely, and the accuracies of both the classical and quantum algorithms are the same. Therefore, we are only going to see an advantage when we restrict the number of vertices that are allowed to have degree close to the maximum (which will be $O(\text{polylog}\, N)$ by necessity). In general, in an effort to make $\|A\| = o(d)$, we should only allow at most $O(\log N)$ vertices to have degree close to the maximum, and the others must have asymptotically smaller (e.g. constant) degree. A class of graphs that satisfies this requirement is the class of power law graphs.

A distribution on power-law graphs is given in [7] for which $d, \bar{d}$ and β are parameters that can be varied freely. In graphs of this type, the number of vertices with degree k is proportional to $k^{-\beta}$, and d is the maximum expected degree of a vertex in the graph, while $\bar{d}$ is the average degree. We have the following results, also from [7]:

1. For $\beta > 3$ and $d > \bar{d}^2 \log^{3+\epsilon} N$, the largest eigenvalue of the graph is almost surely $(1+o(1))\sqrt{d}$, for some $\epsilon = O(1)$, and where $\bar{d}$ denotes the average degree.
2. For $2.5 < \beta < 3$ and $d > \bar{d}^{(\beta-2)/(\beta-2.5)} \log^{3/(\beta-2.5)} N$, the largest eigenvalue of the graph is almost surely $(1+o(1))\sqrt{d}$.
3. For $2 < \beta < 2.5$ and $m > \log^{3/2.5-\beta} N$, the largest eigenvalue is almost surely $(1+o(1))\tilde{d}$.

Note that in all of the above, the bounds still apply when the graph is sparse (i.e. $d = O(\text{polylog}\, N)$). Hence, for power law graphs with exponent $\beta > 2.5$, we almost always get a quadratic improvement in accuracy over the classical algorithm. As the exponent decreases, so does the advantage gained by the quantum algorithm.

Some interesting subclasses of power law graphs have exponents between 2 and 2.5. For example, 'internet graphs' have exponents between 2.1 and 2.4, and the 'Hollywood' graph has exponent ≈ 2.3 [8]. In these cases, we might expect some quantum improvement over a classical approach, but not the full square root improvement.

References

1 D. Aharonov, V. Jones, and Z. Landau. A polynomial quantum algorithm for approximating the Jones polynomial. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 427–436. ACM, 2006. `arXiv:quant-ph/0511096`.

2 D. Aharonov and T. Naveh. Quantum NP-a survey. `arXiv:quant-ph/0210077`, 2002.

3 A. Ben-Aroya, O. Regev, and R. de Wolf. A hypercontractive inequality for matrix-valued functions with applications to quantum computing and LDCs. In *FOCS'08*, pages 477–486. IEEE, 2008. `arXiv:0705.3806`.

4 D. Berry, G. Ahokas, R. Cleve, and B. Sanders. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007. `arXiv:quant-ph/0508139`.

5 D. W Berry, A. Childs, and R. Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 792–809. IEEE, 2015. `arXiv:1312.1414`.

6 F. Brandão. Entanglement theory and the quantum simulation of many-body physics. `arXiv:0810.0026`, 2008.

7 F. Chung, L. Lu, and V. Vu. Spectra of random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 100(11):6313–6318, 2003.

8 Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM computer communication review*, volume 29, pages 251–262. ACM, 1999.

9 O. Goldreich. On promise problems: A survey. In *Theoretical computer science*, pages 254–290. Springer, 2006.

10 I. Gutman. The energy of a graph: old and new results. In *Algebraic combinatorics and applications*, pages 196–211. Springer, 2001.

11 A. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009. `arXiv:0811.3171`.

12 P. Hayden and A. Winter. Counterexamples to the maximal p-norm multiplicativity conjecture for all $p > 1$. *Communications in mathematical physics*, 284(1):263–280, 2008. `arXiv:0807.4753`.

13 D. Janzing and P. Wocjan. BQP-complete problems concerning mixing properties of classical random walks on sparse graphs. `arXiv:quant-ph/0610235`, 2006.

14 D. Janzing and P. Wocjan. A Simple PromiseBQP-complete Matrix Problem. *Theory of computing*, 3(1):61–79, 2007.

15 J. Kempe, A. Kitaev, and O. Regev. The complexity of the local Hamiltonian problem. *SIAM Journal on Computing*, 35(5):1070–1097, 2006.

16 A. Kitaev, A. Shen, and M. Vyalyi. *Classical and quantum computation*, volume 47. American Mathematical Society Providence, 2002.

17 E. Knill and R. Laflamme. Power of one bit of quantum information. *Physical Review Letters*, 81(25):5672, 1998. `arXiv:quant-ph/9802037`.

18 E. Knill and R. Laflamme. Quantum computing and quadratically signed weight enumerators. *Information Processing Letters*, 79(4):173–179, 2001. `arXiv:quant-ph/9909094`.

19 X. Li, Y. Shi, and I. Gutman. *Graph energy*. Springer Science & Business Media, 2012.

20 S. Lloyd. Universal quantum simulators. *Science*, 273(5278):1073, 1996. `quant-ph/9703054`.

21 T. Morimae. Hardness of classically sampling one clean qubit model with constant total variation distance error. `arXiv:1704.03640`, 2017.

22 T. Morimae, K. Fujii, and Joseph F. Fitzsimons. Hardness of classically simulating the one-clean-qubit model. *Physical review letters*, 112(13):130502, 2014. `arXiv:1312.2496`.

23 M. Nielsen and I. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

24 D. Perez-Garcia, M. Wolf, D. Petz, and M. Ruskai. Contractivity of positive and trace-preserving maps under L_p norms. *Journal of Mathematical Physics*, 47(8):083506, 2006. `arXiv:math-ph/0601063`.

25 D. Shepherd. Computation with unitaries and one pure qubit. `arXiv:quant-ph/0608132`, 2006.

26 P. Shor and S. Jordan. Estimating Jones polynomials is a complete problem for one clean qubit. *Quantum Information & Computation*, 8(8):681–714, 2008. `arXiv:0707.2831`.

27 J. Watrous. Quantum computational complexity. In *Encyclopedia of complexity and systems science*, pages 7174–7201. Springer, 2009. `arXiv:0804.3401`.

A Error Analysis of DQC1 Algorithm

A.1 Error from Hamiltonian Simulation

First we consider the error that arises in the circuit from Hamiltonian simulation. We assume that the Hamiltonian simulation step implements a unitary V that approximates e^{iA} in the sense that $||V - e^{iA}|| \leq \delta$, so that the eigenvalues of V and e^{iA} can differ by at most δ. For now, we will assume that the phase estimation routine works perfectly (i.e. introduces no error). Recall that this part of the circuit outputs an estimate for an eigenvalue of A in the range $[-\pi, \pi)$. Denote by λ_j and μ_j the output of the phase estimation routine when it is run using e^{iA} and V, respectively. We have

$$\left|e^{i\lambda_j} - e^{i\mu_j}\right| \leq \delta$$

by the bound on the error of the Hamiltonian simulation, where we can assume $|\mu_j - \lambda_j| \leq \pi$, by adding multiples of 2π to λ_j if necessary. The left hand side can be written as

$$\begin{aligned}
\left|1 - e^{i(\mu_j - \lambda_j)}\right| &= \left|e^{i\frac{(\mu_j-\lambda_j)}{2}}\left(e^{-i\frac{(\mu_j-\lambda_j)}{2}} - e^{i\frac{(\mu_j-\lambda_j)}{2}}\right)\right| \\
&= \left|e^{-i\frac{(\mu_j-\lambda_j)}{2}} - e^{i\frac{(\mu_j-\lambda_j)}{2}}\right| \\
&= 2\left|\sin\left(\frac{\mu_j-\lambda_j}{2}\right)\right| = 2\sin\left|\frac{\mu_j-\lambda_j}{2}\right| \qquad \text{(since } |\mu_j - \lambda_j| \leq 2\pi).
\end{aligned}$$

We will use the inequality $(2/\pi)\theta \leq \sin\theta$ for $0 \leq \theta \leq \pi/2$. Therefore, we have that

$$(4/\pi)\frac{|\mu_j - \lambda_j|}{2} \leq 2\sin\left|\frac{\mu_j-\lambda_j}{2}\right| \leq \delta$$

and hence

$$|\mu_j - \lambda_j| \leq \pi\delta/2.$$

To see how this affects the accuracy of the algorithm, we consider the difference in the trace of U' when using V in place of e^{iA}.

$$\begin{aligned}
2\left|\sum_j f(\lambda_j) - \sum_j f(\mu_j)\right| &\leq 2\sum_j |f(\lambda_j) - f(\mu_j)| \\
&\leq 2\sum_j K\,|\lambda_j - \mu_j| \qquad \text{by the Lipschitz condition} \\
&\leq 2\sum_j K\pi\delta/2 = 2^n K\pi\delta.
\end{aligned}$$

Choosing the simulation accuracy to be $\delta \leq \epsilon/(2\pi)$, this contributes an error term of $2^n\epsilon K/2$. Thus, we have

$$2\left|\sum_j f(\lambda_j) - \sum_j f(\mu_j)\right| \leq 2^n\epsilon K/2. \qquad (1)$$

A.2 Error from Phase Estimation

Here we consider the error that arises from using the phase estimation routine to estimate the eigenvalues μ_j of the unitary V from the previous sub-section. The phase estimation routine requires the addition of a ancilla qubits, which are used to control the application of powers of V on an n-qubit register. More precisely, the lth ancilla qubit is used to control the application of the unitary V^{2^l}, so that we apply the controlled gate

$$W_l := |0\rangle\langle 0|_l \otimes I + |1\rangle\langle 1|_l \otimes V^{2^l}$$

where the subscript l denotes that the projector acts on the lth ancilla/control qubit (and as the identity everywhere else). Let $W := W_1 W_2 \cdots W_a$. Then the phase estimation routine consists of applying Hadamard gates to all of the control qubits, applying W, and then applying the inverse quantum Fourier transform to the control qubits.

If we apply phase estimation to an eigenvector of V with eigenvalue $e^{i2\pi\theta}$, and measure the control register, we obtain some output $x \in \{0, 1, ..., 2^a - 1\}$ such that

$$\Pr(|\theta - x/2^a| < \eta) > 1 - \varphi \tag{2}$$

for $\varphi, \eta > 0$. To obtain this level of accuracy and probability of failure, it is sufficient [23] to set

$$a = \lceil \log(1/\eta) \rceil + \lceil \log(2 + (1/(2\varphi))) \rceil. \tag{3}$$

Let ϕ be defined as follows:

$$\phi(x) := \begin{cases} x2\pi/2^a & \text{if } x \leq 2^{a-1} \\ x2\pi/2^a - 2\pi & \text{otherwise} \end{cases}$$

Then let $\phi(x_j)$ be our estimate of the eigenvalue μ_j corresponding to the eigenvector $|\psi_j\rangle$, which, by the definition of ϕ above, lies in the interval $[-\pi, \pi)$. By Equation (2), if we apply phase estimation to an eigenvector $|\psi_j\rangle$ of V with corresponding eigenvalue $e^{i\mu_j}$, and measure, we have

$$\Pr(|\mu_j - \phi(x_j)| < 2\pi\eta) > 1 - \varphi \tag{4}$$

where the extra factor of 2π results from rescaling the value of x_j by 2π.

In our case, we do not measure the control register, and therefore we do not collapse the superposition over eigenvalues that phase estimation produces. Here we consider the effect that this has on the output of the algorithm, and simultaneously bound the error introduced by this part of the circuit. When phase estimation does not work perfectly, the algorithm consists of the following steps, implementing a unitary $\tilde{U}$:

1. Apply phase estimation on $V \approx e^{iA}$ with the input $|\psi_j\rangle$, to obtain a superposition over estimates $\phi(k)$ of the eigenvalue $\mu_j = 2\pi\theta_j$:
$$|\psi_j\rangle (\alpha |0\rangle + \beta |1\rangle) |0 \ldots 0\rangle \mapsto |\psi_j\rangle (\alpha |0\rangle + \beta |1\rangle) \sum_k \gamma_{k|j} |\phi(k)\rangle$$
where $\gamma_{k|j} = \frac{1}{N} \sum_a e^{2\pi i a(\theta_j - k/N)}$.
2. Perform controlled phase rotations:
$$|\psi_j\rangle (\alpha |0\rangle + \beta |1\rangle) \sum_k \gamma_{k|j} |\phi(k)\rangle$$
$$\mapsto |\psi_j\rangle \sum_k \gamma_{k|j} (\alpha e^{i \arccos(f(\phi(k)))} |0\rangle + \beta e^{-i \arccos(f(\phi(k)))} |1\rangle) |\phi(k)\rangle .$$

3. Undo the phase estimation to uncompute the value in the 3rd register. To undo phase estimation we: a) apply the QFT to the register containing the $\phi(k)$'s, b) apply controlled powers of the unitary $V^\dagger \approx e^{-iA}$, and c) apply Hadamard gates to all qubits in the third register.
   a. Apply the QFT:

$$|\psi_j\rangle \frac{1}{\sqrt{N}} \sum_k \gamma_{k|j} (\alpha e^{i \arccos(f(\phi(k)))} |0\rangle + \beta e^{-i \arccos(f(\phi(k)))} |1\rangle) \sum_w e^{2\pi i w k/N} |w\rangle .$$

   b. Apply the controlled (on the third register) $V^\dagger$ gates:

$$|\psi_j\rangle \frac{1}{\sqrt{N}} \sum_k \gamma_{k|j} (\alpha e^{i \arccos(f(\phi(k)))} |0\rangle + \beta e^{-i \arccos(f(\phi(k)))} |1\rangle) \sum_w e^{2\pi i w k/N} e^{-2\pi i \theta_j w} |w\rangle .$$

   c. Apply Hadamard gates to each of the ancilla qubits:

$$\begin{aligned} &|\psi_j\rangle \frac{1}{N} \sum_k \gamma_{k|j} (\alpha e^{i \arccos(f(\phi(k)))} |0\rangle \\ &\qquad + \beta e^{-i \arccos(f(\phi((k)))} |1\rangle) \sum_w \sum_x e^{2\pi i w k/N} e^{-2\pi i \theta_j w} (-1)^{w \cdot x} |x\rangle . \end{aligned}$$

This means that $\tilde{U}$ performs the mapping

$$\begin{aligned} &|\psi_j\rangle (\alpha |0\rangle + \beta |1\rangle) |0 \ldots 0\rangle \\ &\mapsto |\psi_j\rangle \frac{1}{N} \sum_k \gamma_{k|j} (\alpha e^{i \arccos(f(\phi(k)))} |0\rangle \\ &\qquad + \beta e^{-i \arccos(f(\phi(k)))} |1\rangle) \sum_x \left(\sum_w e^{2\pi i w k/N} e^{-2\pi i \theta_j w} (-1)^{w \cdot x} \right) |x\rangle \end{aligned}$$

for each eigenvector $|\psi_j\rangle$ of V.

Let $\{|\psi_j\rangle |b\rangle |\phi\rangle, b \in \{0,1\}\}$ be a basis for the tensor product of the three registers. By design, the only states that contribute to the trace of U' are those of the form $|\psi_j\rangle |b\rangle |0 \ldots 0\rangle$. Hence, we can consider the trace of $\tilde{U'}$ – the submatrix of $\tilde{U}$ in which the third register is in the state $|0 \ldots 0\rangle$) – which is given by:

$$\begin{aligned} \mathrm{Tr}(\tilde{U'}) &= \sum_j \langle\psi_j| \langle 0| \left(|\psi_j\rangle \frac{1}{N} \sum_k \gamma_{k|j} \sum_w e^{2\pi i w k/N} e^{-2\pi i \theta_j w} e^{i \arccos(f(\phi(k)))} |0\rangle \right) \\ &+ \sum_j \langle\psi_j| \langle 1| \left(|\psi_j\rangle \frac{1}{N} \sum_k \gamma_{k|j} \sum_w e^{2\pi i w k/N} e^{-2\pi i \theta_j w} e^{-i \arccos(f(\phi(k)))} |1\rangle \right) \\ &= \frac{1}{N} \sum_{j,k} \gamma_{k|j} \sum_w e^{2\pi i w k/N} e^{-2\pi i \theta_j w} \left(e^{i \arccos(f(\phi(k)))} + e^{-i \arccos(f(\phi(k)))} \right) \\ &= \frac{1}{N} \sum_{j,k} \gamma_{k|j} 2 f(\phi(k)) \sum_w e^{2\pi i w (k/N - \theta_j)} \\ &= 2 \sum_{j,k} \left|\gamma_{k|j}\right|^2 f(\phi(k)) \\ &= 2 \sum_k f(\phi(k)) \sum_j \left|\gamma_{k|j}\right|^2 . \end{aligned}$$

Suppose that $\theta_j = z_j/N$ for some z_j – that is, each θ_j can be represented precisely by an n-bit rational number z_j/N. Then $\gamma_{k|j} = \delta_{k,z_j}$, and so $\mathrm{Tr}(\tilde{U}') = 2\sum_j f(\mu_j)$. This corresponds to the case in which phase estimation works perfectly; in reality, we will not be able to express all eigenvalues precisely as n-bit rational numbers. Instead, suppose that $\theta_j = \tilde{z}_j/N + \delta_j$, where $\tilde{z}_j/N$ is the closest n-bit approximation of θ_j, and so $0 \leq \delta_j \leq 1/(2N)$. The difference between the trace in the two cases is given by

$$\begin{aligned} 2\left|\sum_j f(\mu_j) - \sum_j\sum_k \left|\gamma_{k|j}\right|^2 f(\phi(k))\right| &\leq 2\sum_j \left| f(\mu_j) - \sum_k \left|\gamma_{k|j}\right|^2 f(\phi(k))\right| \\ &= 2\sum_j\left|\sum_k \left|\gamma_{k|j}\right|^2 (f(\mu_j) - f(\phi(k)))\right| \\ &\leq 2\sum_j\sum_k \left|\gamma_{k|j}\right|^2 \left|f(\mu_j) - f(\phi(k))\right|, \end{aligned}$$

where the second step follows because $\sum_k \left|\gamma_{k|j}\right|^2 = 1$. The coefficient $\left|\gamma_{k|j}\right|^2$ is precisely the probability of measuring $\phi(k)$ on the ancilla register when the true eigenvalue is μ_j. By the promises of phase estimation (Equation (4)), with probability $\leq \varphi$ we have $|\mu_j - \phi(k)| > 2\pi\eta$, in which case $|f(\mu_j) - f(\phi(k))| \leq 2f_{\max}$; and with probability $\geq 1-\varphi$ we have $|\mu_j - \phi(k)| \leq 2\pi\eta$, in which case $|f(\mu_j) - f(\phi(k))| \leq 2\pi K\eta$. Hence, the error from this part of the circuit is bounded above by

$$2\left|\sum_j f(\mu_j) - \sum_j\sum_k \left|\gamma_{k|j}\right|^2 f(\phi(k))\right| \leq 4\sum_j(\pi K\eta + \varphi f_{\max}) = 2^{n+2}(\pi K\eta + \varphi f_{\max}).$$

Choosing $\eta < \epsilon/(8\pi)$ and $\varphi < \epsilon/8$, and assuming that $f_{\max} \leq 1$ (as stated earlier), this becomes

$$2\left|\sum_j f(\mu_j) - \sum_j\sum_k \left|\gamma_{k|j}\right|^2 f(\phi(k))\right| \leq 2^n\frac{1}{2}\epsilon(K+1). \tag{5}$$

Now we consider how this contributes to the overall error. As before, let λ_j denote the eigenvalues of e^{iA}. Then the error of the algorithm, taking into account both the Hamiltonian simulation and phase estimation steps, is

$$\begin{aligned} &2\left|\sum_j f(\lambda_j) - \sum_j\sum_k \left|\gamma_{k|j}\right|^2 f(k)\right| \\ &\leq 2\left|\sum_j f(\lambda_j) - \sum_j f(\mu_j)\right| + 2\left|\sum_j f(\mu_j) - \sum_j\sum_k \left|\gamma_{k|j}\right|^2 f(k)\right| \end{aligned}$$

where the first term on the right corresponds to the error from the Hamiltonian simulation part of the circuit (i.e. the difference between the trace of the circuit when using V instead of e^{iA}), and the second term corresponds to the error introduced by phase estimation. A bound on the first term is given by Equation (1), and the second term is bounded via Equation (5). Therefore, the difference in the trace of U' in the case where Hamiltonian simulation and phase estimation both work perfectly, and when they do not, is bounded by

$$2\left|\sum_j f(\lambda_j) - \sum_j\sum_k \left|\gamma_{k|j}\right|^2 f(k)\right| \leq 2^n\epsilon(K+1/2). \tag{6}$$

A.3 Error from estimating $\mathrm{Tr}(U')/2^n$ in the DQC1 model

The one clean qubit model can estimate the normalised trace of a $2^n \times 2^n$ sub-matrix of a $2^{n+O(\log n)} \times 2^{n+O(\log n)}$ unitary matrix (implemented by a poly(n)-sized circuit) up to accuracy $\zeta = \Omega(1/\operatorname{poly}(n))$. Therefore, using the one clean qubit model to estimate the trace of U' will introduce an extra error term ζ. Let $\widetilde{\mathrm{Tr}(U')}/2^n$ be the output from the one clean qubit algorithm. Then choosing $\zeta = \epsilon/2$, and using the bound from Equation (6), we have

$$\left| \frac{2}{2^n} \sum_j f(\lambda_j) - \widetilde{\mathrm{Tr}(U')}/2^n \right| \leq \epsilon(K+1). \tag{7}$$

Hence, we can estimate $\frac{1}{2^n} \sum_j f(\lambda_j)$ in polynomial time with accuracy $\epsilon(K+1)$ for any $\epsilon = \Omega(1/\operatorname{poly}(n))$.

B Classical Algorithms

Here we describe a classical algorithm for diagonal entry estimation, which is the problem of estimating an entry on the diagonal of the matrix A^p, up to reasonable error. Given the ability to estimate the diagonal entries of a matrix, we are able to estimate the normalised trace of that matrix.

We first present an algorithm for the special case where A contains only $0, 1$ entries, and then in Appendix B.1 discuss how it can be extended to work for arbitrary real matrices. In the first case, the matrix A defines an unweighted, undirected graph with N vertices. The value of $(A^p)_{jj}$ is equivalent to the number of distinct walks (i.e. traversals around the graph that may traverse any edge more than once, or not at all) of length p starting and ending at vertex j.

We begin by observing that $(A^p)_{jj}$ can be re-interpreted as the total number of walks of length p leaving j multiplied by the probability that such a walk ends at vertex j. We can obtain an estimate of the latter by performing a number of random walks of length p, beginning at vertex j, and counting how many of them return to vertex j on the final step.

In order to obtain an estimate of the total number of walks of length p leaving a given vertex, we can do the following: given an upper bound d on the degree of the graph, we generate a number of sequences of p integers chosen independently and uniformly at random from the range $[0, d]$. Any given sequence provides a ‘candidate’ walk of length p on the graph, which may or may not be realisable on the graph defined by A. Given a candidate walk of the form $(n_0, n_1, ..., n_p)$, we test whether or not it is realisable by starting a walk at vertex j, and then moving to the n_0th neighbour of j. We then move to the n_1th neighbour of that vertex, and so on. If, at any step i of the walk, a vertex does not have a neighbour n_i, we terminate the process and conclude that the candidate is not realisable.

If we tried all d^p possible candidate walks from vertex j, then by counting the number of successes we would know the exact value of the number of walks of length p that leave vertex j; however, this would require $O(d^p)$ walks to be performed. If instead we sample from the set of all possible walks by generating a number of sequences at random, we can obtain a close estimate of the true number of walks. Below is the full algorithm for diagonal entry estimation. We assume that we are given some bound d on the degree of the graph, and that we wish to estimate $(A^p)_{jj}$.

1. Estimate the total number of walks of length p leaving vertex j:
   a. Define variables X_i for $i \in [k]$, for some value of k to be determined later.

   b. For $i = 1$ to k:
      i. Generate a sequence $(n_0, n_1, ..., n_p)$, where each $n_l \in [d]$.
      ii. Attempt to follow the walk defined by the sequence.
      iii. If the walk was successful, set $X_i = 1$, otherwise set $X_i = 0$.
   c. Then $\overline{X} = \frac{d^p}{k}(X_1 + X_2 + ... + X_k)$ provides an estimate of the total number of walks of length p leaving vertex j.
2. Estimate the probability that a given walk returns to vertex j:
   a. Define variables Y_i for $i \in [k']$, for some value of k' to be determined later.
   b. For $i = 1$ to k':
      i. Perform a random walk of length p starting at vertex j.
      ii. If the walk returns to vertex j (as its final step), then set $Y_i = 1$, otherwise set it to 0.
   c. Then $\overline{Y} = \frac{1}{k'}(Y_1 + Y_2 + ... + Y_{k'})$ gives an estimate of the probability that a given walk returns to vertex j.
3. Multiplying the two values together gives us our desired estimate: $(\tilde{A^p})_{jj} = \overline{X} \cdot \overline{Y}$.

To analyse the accuracy of this estimation, we will look at the errors in the two estimates $\overline{X}$ and $\overline{Y}$.

In both steps, we are essentially aiming to estimate the success probability of some Bernoulli process: in step 1 we aim to estimate the probability with which a randomly generated sequence of 'moves' succeeds in generating a valid walk around the graph, and in step 2 we are estimating the probability that a given (valid) walk of length p succeeds in returning to its starting vertex on the final step of the walk. In both cases, we can estimate the appropriate probability up any desired accuracy ϵ by choosing the number of samples (k in step 1, and k' in step 2) to be inverse polynomial in ϵ.

We use Hoeffding's inequality to bound the accuracy of both estimates. For step 1, we absorb the factor of d^p into the random variables X_i, and use the general form of the bound:

$$\Pr\left[|\overline{X} - \mathbb{E}[\overline{X}]| \geq \epsilon d^p\right] \leq 2e^{-2\epsilon^2 k}.$$

And for step 2, we have

$$\Pr\left[|\overline{Y} - \mathbb{E}[\overline{Y}]| \geq \epsilon'\right] \leq 2e^{-2\epsilon'^2 k'}.$$

Therefore, by choosing $k = \operatorname{poly}(1/\epsilon)$ and $k' = \operatorname{poly}(1/\epsilon')$, we can estimate $(A^p)_{jj}$ up to additive error that is at most $d^p(\epsilon + \epsilon' + \epsilon\epsilon) = d^p\delta$ for $\delta = 1/\operatorname{poly}(n)$, with a constant probability of failure.

B.1 Extension to real matrices

In this section we extend the diagonal entry estimation algorithm of the previous section to work for arbitrary real matrices. Recall that this algorithm works for matrices with $0, 1$ entries by interpreting the input matrix as the adjacency matrix for an unweighted, undirected graph. More general (symmetric) matrices may be viewed as undirected graphs with weighted edges, and a similar interpretation of the value of $(A^p)_{jj}$ holds in these cases.

In the case of general matrices, the value of $(A^p)_{jj}$ depends not only on the number of closed walks (i.e. those that return to their start vertex) leaving vertex j, but also on the 'weight' of those walks. Let $\mathcal{C}_p^j$ be the set of all *closed* walks of length p leaving vertex j, and $E(\omega)$ be the set of edges that make up a given walk ω.

Then we have

$$(A^p)_{jj} = \sum_{c \in \mathcal{C}_p^j} \prod_{e \in E(c)} \text{weight}(e).$$

In order to estimate this quantity, we proceed similarly to the above case.

Let us denote the set of all (not necessarily closed) walks of length p originating at vertex j by $\mathcal{W}_p^j$. Then we can re-write the above quantity as

$$(A^p)_{jj} = W_p \, \mathbb{E}_{\omega \in \mathcal{W}_p^j} \left[\prod_{e \in E(\omega)} \text{weight}(e) \right],$$

by using the same reasoning as before – i.e. that the jth diagonal entry of A^p is given by the total number of walks of length p leaving vertex j multiplied by the expected 'weight' of each walk, where we assign a weight of 0 if the walk does not return to vertex j.

We can estimate the expectation on the right by sampling from the set of closed walks of length p originating at vertex j. This can be done by performing random walks of length p starting at vertex j, and recording the total weights of those walks that return to vertex j. This is easily incorporated into the existing algorithm: we set the variable Y_i to 0 if the ith walk does not return to vertex j, and otherwise we set it to the total weight of the walk (i.e. the product over the weights of the edges of the walk). W_p can be estimated as before, up to error ϵd^p. The error in estimating the expectation value depends upon the largest total weight of a closed walk in the graph. This is smaller than or equal to $\|A\|_{\max}^p$, where $\|A\|_{\max}$ is the maximum absolute size of an entry in A. A bound on the accuracy of estimating the expectation value is once again given by Hoeffding's inequality:

$$\Pr[|\overline{Y} - \mathbb{E}[\overline{Y}]| \geq \epsilon' \|A\|_{\max}^p] \leq 2e^{-2\epsilon'^2 k'}.$$

Multiplying the two estimates together, we obtain an estimate of $(A^p)_{jj}$ up to accuracy $\delta d^p \|A\|_{\max}^p$ with constant probability.

B.2 Estimating $\mathrm{Tr}(A^p)/N$ Classically

We can use the classical version of diagonal entry estimation to estimate the normalised trace of a matrix. More precisely, we obtain the empirical mean of $(A^p)_{jj}$ over a sample of values of j chosen uniformly at random. To see that the mean value of $(A^p)_{jj}$ for $j \in [N]$ does indeed give us the desired value, we observe that

$$\mathbb{E}_j[(A^p)_{jj}] = \frac{1}{N} \sum_{j=0}^{N-1} (A^p)_{jj} = \frac{\mathrm{Tr}(A^p)}{N}.$$

Let the output of the diagonal entry estimation algorithm be $\widetilde{(A^p)}_{jj}$ (which is an estimate of $(A^p)_{jj}$ up to additive error $\delta d^p \|A\|_{\max}^p$). Then let $\overline{\widetilde{(A^p)}_{jj}}$ be the mean value of the variable $\widetilde{(A^p)}_{jj}$ after sampling k times for randomly chosen values of j. The value of $\widetilde{(A^p)}_{jj}$ is bounded in the interval $[-(d\|A\|_{\max})^p, (d\|A\|_{\max})^p]$. Then by Hoeffding's inequality:

$$\Pr\left[\left|\overline{\widetilde{(A^p)}_{jj}} - \mathbb{E}[\widetilde{(A^p)}_{jj}]\right| \geq \delta d^p \|A\|_{\max}^p\right] \leq 2\exp\left(\frac{-\delta^2}{2}k\right).$$

Thus, choosing k to be inverse polynomial in δ allows us to obtain an estimate of $\mathbb{E}[(A^p)_{jj}] = \mathrm{Tr}(A^p)/N$ up to error $\delta d^p \|A\|_{\max}^p$. Note that for $0, 1$ and $-1, 0, +1$ matrices, $\|A\|_{\max} = 1$ and therefore the accuracy of the estimation in this case is just δd^p.