Quantum algorithm for linear systems of equations

Aram W. Harrow*, Avinatan Hassidim†and Seth Lloyd‡

June 2, 2009

Abstract

Solving linear systems of equations is a common problem that arises both on its own and as a subroutine in more complex problems: given a matrix A and a vector $\vec{b}$, find a vector $\vec{x}$ such that $A\vec{x} = \vec{b}$. We consider the case where one doesn't need to know the solution $\vec{x}$ itself, but rather an approximation of the expectation value of some operator associated with $\vec{x}$, e.g., $\vec{x}^\dagger M\vec{x}$ for some matrix M. In this case, when A is sparse and well-conditioned, with largest dimension N, the best classical algorithms can find $\vec{x}$ and estimate $\vec{x}^\dagger M\vec{x}$ in $O(N \operatorname{poly}\log(N))$ time. Here, we exhibit a quantum algorithm for this task that runs in $\operatorname{poly}(\log N)$ time, an exponential improvement over the best classical algorithm.

Quantum computers are devices that harness quantum mechanics to perform computations in ways that classical computers cannot. For certain problems, quantum algorithms supply exponential speedups over their classical counterparts, the most famous example being Shor's factoring algorithm [1]. Few such exponential speedups are known, and those that are (such as the use of quantum computers to simulate other quantum systems [2]) have found little use outside the domain of quantum information theory. This paper presents a quantum algorithm that can give an exponential speedup for a broad range of applications.

Linear equations play an important role in virtually all fields of science and engineering. The sizes of the data sets that define the equations are growing rapidly over time, so that terabytes and even petabytes of data may need to be processed to obtain a solution. The minimum time it takes to exactly solve such a set on a classical computer scales at least as N, where N is the size of the data set. Indeed, merely to write out the solution takes time of order N. Frequently, however, one is interested not in the full solution to the equations, but rather in computing some function of that solution, such as determining the total weight of some subset of the indices. We show that in some cases, a

*Department of Mathematics, University of Bristol, Bristol, BS8 1TW, U.K.

†MIT - Research Laboratory for Electronics, Cambridge, MA 02139, USA

‡MIT - Research Laboratory for Electronics and Department of Mechanical Engineering, Cambridge, MA 02139, USA

quantum computer can approximate the value of such a function in time which is polylogarithmic in N, an exponential speedup over the best known classical algorithms. In fact, under standard complexity-theoretic assumptions, we prove that in performing this task any classical algorithm must be exponentially slower than the algorithm presented here. Moreover, we show that our algorithm is almost the optimal quantum algorithm for the task.

We begin by presenting the main ideas behind the construction. Then we give an informal description of the algorithm, making many simplifying assumptions. Finally we present generalizations and extensions. The full proofs appear in the supporting online material [3].

Assume we are given the equation $A\vec{x} = \vec{b}$, where $\vec{b}$ has N entries. The algorithm works by mapping $\vec{b}$ to a quantum state $|b\rangle$ and by mapping A to a suitable quantum operator. For example, A could represent a discretized differential operator which is mapped to a Hermitian matrix with efficiently computable entries, and $|b\rangle$ could be the ground state of a physical system, or the output of some other quantum computation. Alternatively, the entries of A and $\vec{b}$ could represent classical data stored in memory. The key requirement here, as in all quantum information theory, is the ability to perform actions in superposition (also called "quantum parallel"). We present an informal discussion of superposition, and its meaning in this context. Suppose that a algorithm (which we can take to be reversible without loss of generality) exists to map input $x, 0$ to output $x, f(x)$. Quantum mechanics predicts that given a superposition of $x, 0$ and $y, 0$, evaluating this function on a quanutm computer will produce a superposition of $x, f(x)$ and $y, f(y)$, while requiring no extra time to execute. One can view accessing a classical memory cell as applying a function whose input is the address of the cell and which output the contents of this cell. We require that we can access this function in superposition.

In the following paragraphs we assume that A is sparse and Hermitian. Both assumptions can be relaxed, but this complicates the presentation. We also ignore some normalization issues (which are treated in the supplementary material). The exponential speedup is attained when the condition number of A is polylogarithmic in N, and the required accuracy is $1/\operatorname{poly}\log n$.

The algorithm maps the N entries of $\vec{b}$ onto the $\log_2 N$ qubits required to represent the state $|b\rangle$. When A is sparse, the transformation $e^{iAt}|b\rangle$ can be implemented efficiently. This ability to exponentiate A translates, via the well-known technique of phase estimation, into the ability to decompose $|b\rangle$ in the eigenbasis of A and to find the corresponding eigenvalues λ_j. Informally, the state of the system after this stage is close to $\sum_j \beta_j |u_j\rangle |\lambda_j\rangle$, where u_j is the eigenvector basis of A, and $|b\rangle = \sum_j \beta_j |u_j\rangle$. As the eigenvalues which corresponds to each eigenvector is entangled with it, One can hope to apply an operation which would take $\sum_j \beta_j |u_j\rangle |\lambda_j\rangle$ to $\sum_j \lambda_j^{-1} \beta_j |u_j\rangle |\lambda_j\rangle$. However, this is not a linear operation, and therefore performing it requires a unitary, followed by a successful measurement. This allows us to extract the state $|x\rangle = A^{-1}|b\rangle$. The total number of resources required to perform these transformations scales poly-logarithmically with N.

This procedure yields a quantum-mechanical representation $|x\rangle$ of the desired vector $\vec{x}$. Clearly, to read out all the components of $\vec{x}$ would require one to perform the procedure at least N times. However, when one is interested not in $\vec{x}$ itself, but in some expectation value $\vec{x}^T M \vec{x}$, where M is some linear operator (our procedure also accommodates nonlinear operators as described below). By mapping M to a quantum-mechanical operator, and performing the quantum measurement corresponding to M, we obtain an estimate of the expectation value $\langle x| M |x\rangle = \vec{x}^T M \vec{x}$, as desired. A wide variety of features of the vector $\vec{x}$ can be extracted in this way, including normalization, weights in different parts of the state space, moments, etc.

A simple example where the algorithm can be used is to see if two different stochastic processes have similar stable state [4]. Consider a stochastic process $x_t = Ax_{t-1} + b$, where the i'th coordinate in the vector x_t represents the abundance of item i in time t. The stable state of this distribution is given by $|x\rangle = (I - A)^{-1} |b\rangle$. Let $\tilde{x}_t = \tilde{A}\tilde{x}_{t-1} + \tilde{b}$, and $|\tilde{x}\rangle = (I - \tilde{A})^{-1} \left|\tilde{b}\right\rangle$. To know if $|x\rangle$ and $|\tilde{x}\rangle$ are similar, we perform the SWAP test between them [5]. We note that classically finding out if two probability distributions are similar requires at least $O(\sqrt{N})$ samples [6]. One can apply similar ideas to know if different pictures are similar, or to identify what is the relation between two pictures. In general, different problems require us to extract different features, and it is an important question to identify what are the important features to extract.

Estimating expectation values on solutions of systems of linear equations is quite powerful. In particular, we show that it is universal for quantum computation - anything that a quantum computer can do, can be written as a set of linear equations, such that the result of the computation is encoded in some expectation value of the solution of the system. Thus, matrix inversion can be thought of as an alternate paradigm for quantum computing, along with [7, 8, 9, 10, 11, 12]. Matrix inversion has the advantage of being a natural problem that is not obviously related to quantum mechanics. We use the universality result to show that our algorithm is almost optimal and that classical algorithms cannot match its performance.

An important factor in the performance of the matrix inversion algorithm is κ, the condition number of A, or the ratio between A's largest and smallest eigenvalues. As the condition number grows, A becomes closer to a matrix which cannot be inverted, and the solutions become less stable. Such a matrix is said to be "ill-conditioned." Our algorithms will generally assume that the singular values of A lie between $1/\kappa$ and 1; equivalently $\kappa^{-2} I \leq A^\dagger A \leq I$. In this case, we will achieve a runtime proportional to $\kappa^2 \log N$. However, we also present a technique to handle ill-conditioned matrices. The run-time also scales as $1/\epsilon$ if we allow an additive error of ϵ in the output state $|x\rangle$. Therefore, if κ and $1/\epsilon$ are both $\operatorname{poly}\log(N)$, the run-time will also be $\operatorname{poly}\log(N)$. In this case, our quantum algorithm is exponentially faster than any classical method.

Previous papers utilized quantum computers to perform linear algebraic operations in a limited setting [13]. Our work was extended by [14] to solving nonlinear differential equations.

We now give a more detailed explanation of the algorithm. First, we want to transform a given Hermitian matrix A into a unitary operator e^{iAt} which we can apply at will. This is possible (for example) if A is s-sparse and efficiently row computable, meaning it has at most s nonzero entries per row and given a row index these entries can be computed in time $O(s)$. Under these assumptions, Ref. [15] shows how to simulate e^{iAt} in time

$$\tilde{O}(\log(N)s^2t),$$

where the $\tilde{O}$ suppresses more slowly-growing terms (included in the supporting material [3]). If A is not Hermitian, define

$$C = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix} \tag{1}$$

As C is Hermitian, we can solve the equation $C\vec{y} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}$ to obtain $y = \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix}$. Applying this reduction if necessary, the rest of the paper assumes that A is Hermitian.

We also need an efficient procedure to prepare $|b\rangle$. For example, if b_i and $\sum_{i=i_1}^{i_2} |b_i|^2$ are efficiently computable then we can use the procedure of Ref. [16] to prepare $|b\rangle$.

The next step is to decompose $|b\rangle$ in the eigenvector basis, using phase estimation [17, 18]. Denote by $|u_j\rangle$ the eigenvectors of e^{iAt}, and by λ_j the corresponding eigenvalues. Let

$$|\Psi_0\rangle := \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin \frac{\pi(\tau + \frac{1}{2})}{T} |\tau\rangle \tag{2}$$

for some large T. The coefficients of $|\Psi_0\rangle$ are chosen (following [18]) to minimize a certain quadratic loss function which appears in our error analysis (see supplementary material for details).

Next we apply the conditional Hamiltonian evolution $\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau|^C \otimes e^{iA\tau t_0/T}$ on $|\Psi_0\rangle^C \otimes |b\rangle$, where $t_0 = O(\kappa/\epsilon)$. Fourier transforming the first register gives the state

$$\sum_{j=1}^{N} \sum_{k=0}^{T-1} \alpha_{k|j}\beta_j |k\rangle |u_j\rangle , \tag{3}$$

where $|k\rangle$ are the Fourier basis states, and $|\alpha_{k|j}|$ is large if and only if $\lambda_j \approx \frac{2\pi k}{t_0}$. Defining $\tilde{\lambda}_k := 2\pi k/t_0$, we can relabel our $|k\rangle$ register to obtain

$$\sum_{j=1}^{N} \sum_{k=0}^{T-1} \alpha_{k|j}\beta_j \left|\tilde{\lambda}_k\right\rangle |u_j\rangle$$

Adding an ancilla qubit and rotating conditioned on $\left|\tilde{\lambda}_k\right\rangle$ yields

$$\sum_{j=1}^{N} \sum_{k=0}^{T-1} \alpha_{k|j}\beta_j \left|\tilde{\lambda}_k\right\rangle |u_j\rangle \left(\sqrt{1 - \frac{C^2}{\tilde{\lambda}_k^2}} |0\rangle + \frac{C}{\tilde{\lambda}_k} |1\rangle\right),$$

where $C = O(1/\kappa)$. We now undo the phase estimation to uncompute the $\left|\tilde{\lambda}_k\right\rangle$. If the phase estimation were perfect, we would have $\alpha_{k|j} = 1$ if $\tilde{\lambda}_k = \lambda_j$, and 0 otherwise. Assuming this for now, we obtain

$$\sum_{j=1}^{N} \beta_j \left|u_j\right\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} \left|0\right\rangle + \frac{C}{\lambda_j} \left|1\right\rangle\right)$$

To finish the inversion we measure the last qubit. Conditioned on seeing 1, we have the state

$$\sqrt{\frac{1}{\sum_{j=1}^{N} C^2/|\lambda_j|^2}} \sum_{j=1}^{N} \beta_j \frac{C}{\lambda_j} \left|u_j\right\rangle$$

which corresponds to $\left|x\right\rangle = \sum_{j=1}^{n} \beta_j \lambda_j^{-1} \left|u_j\right\rangle$ up to normalization. We can determine the normalization constant from the probability of obtaining 1. Finally, we make a measurement M whose expectation value $\left\langle x\right| M \left|x\right\rangle$ corresponds to the feature of $\vec{x}$ that we wish to evaluate.

We present an informal description of the sources of error; the exact error analysis and runtime considerations are presented in [3]. Performing the phase estimation is done by simulating e^{iAt}. Assuming that A is s-sparse, this can be done with negligible error in time nearly linear in t and quadratic in s.

The dominant source of error is phase estimation. This step errs by $O(1/t_0)$ in estimating λ, which translates into a relative error of $O(1/\lambda t_0)$ in λ^{-1}. If $\lambda \geq 1/\kappa$ taking $t_0 = O(\kappa/\epsilon)$ induces a final error of ϵ. Finally, we consider the success probability of the post-selection process. Since $C = O(1/\kappa)$ and $\lambda \leq 1$, this probability is at least $\Omega(1/\kappa^2)$. Using amplitude amplification [19], we find that $O(\kappa)$ repetitions are sufficient. Putting this all together, the runtime is

$$\tilde{O}\left(\log(N) s^2 \kappa^2 / \epsilon\right)$$

By contrast, one of the best general-purpose classical matrix inversion algorithms is the conjugate gradient method [20], which, when A is positive definite, uses $O(\sqrt{\kappa}\log(1/\epsilon))$ matrix-vector multiplications each taking time $O(Ns)$ for a total runtime of $O(Ns\sqrt{\kappa}\log(1/\epsilon))$. (If A is not positive definite, $O(\kappa\log(1/\epsilon))$ multiplications are required, for a total time of $O(Ns\kappa\log(1/\epsilon))$.) An important question is whether classical methods can be improved when only a summary statistic of the solution, such as $\vec{x}^\dagger M \vec{x}$, is required. Another question is whether our quantum algorithm could be improved, say to achieve error ϵ in time proportional to $\operatorname{poly}\log(1/\epsilon)$. We show that the answer to both questions is negative, using an argument from complexity theory. Our strategy is to prove that the ability to invert matrices (with the right choice of parameters) can be used to simulate a general quantum computation.

We show that a quantum circuit using n qubits and T gates can be simulated by inverting an $O(1)$-sparse matrix A of dimension $N = O(2^n\kappa)$. The condition number κ is $O(T^2)$ if we need A to be positive definite or $O(T)$ if not. This implies that a classical $\operatorname{poly}(\log N, \kappa, 1/\epsilon)$-time algorithm would be able to simulate a $\operatorname{poly}(n)$-gate quantum algorithm in $\operatorname{poly}(n)$ time. Such a simulation is

strongly conjectured to be false, and is known to be impossible in the presence of oracles [21].

The reduction from a general quantum circuit to a matrix inversion problem, also implies that our algorithm cannot be substantially improved (under standard assumptions). If the run-time could be made polylogarithmic in κ, then any problem solvable on n qubits could be solved in $\mathrm{poly}(n)$ time (i.e. **BQP**=**PSPACE**), a highly implausible result[22]. Even improving our κ-dependence to $\kappa^{1-\delta}$ for $\delta > 0$ would allow any time-T quantum algorithm to be simulated in time $o(T)$; iterating this would again imply that **BQP**=**PSPACE**. Similarly, improving the error dependence to $\mathrm{poly}\log(1/\epsilon)$ would imply that **BQP** includes **PP**, and even minor improvements would contradict oracle lower bounds [22].

We now present the key reduction from simulating a quantum circuit to matrix inversion. Let $\mathcal{C}$ be a quantum circuit acting on $n = \log N$ qubits which applies T two-qubit gates $U_1, \ldots U_T$. The initial state is $|0\rangle^{\otimes n}$ and the answer is determined by measuring the first qubit of the final state.

Now adjoin an ancilla register of dimension $3T$ and define a unitary

$$U = \sum_{t=1}^{T} |t+1\rangle\langle t| \otimes U_t + |t+T+1\rangle\langle t+T| \otimes I + |t+2T+1 \bmod 3T\rangle\langle t+2T| \otimes U_{3T+1-t}^{\dagger}. \tag{4}$$

We have chosen U so that for $T+1 \leq t \leq 2T$, applying U^t to $|1\rangle|\psi\rangle$ yields $|t+1\rangle \otimes U_T \cdots U_1 |\psi\rangle$. If we now define $A = I - Ue^{-1/T}$ then $\kappa(A) = O(T)$, and we can expand

$$A^{-1} = \sum_{k \geq 0} U^k e^{-k/T}, \tag{5}$$

This can be interpreted as applying U^t for t a geometrically-distributed random variable. Since $U^{3T} = I$, we can assume $1 \leq t \leq 3T$. If we measure the first register and obtain $T+1 \leq t \leq 2T$ (which occurs with probability $e^{-2}/(1+e^{-2}+e^{-4}) \geq 1/10$) then we are left with the second register in the state $U_T \cdots U_1 |\psi\rangle$, corresponding to a successful computation. Sampling from $|x\rangle$ allows us to sample from the results of the computation. This establishes that matrix inversion is **BQP**-complete, and proves our above claims about the difficulty of improving our algorithm.

We now discuss ways to extend our algorithm and relax the assumptions we made while presenting it. First, we show how a broader class of matrices can be inverted, and then consider measuring other features of $\vec{x}$ and performing operations on A other than inversion.

Certain non-sparse A can also be simulated and therefore inverted; see [23] for a list of examples. It is also possible to invert non-square matrices, using the reduction presented from the non-Hermitian case to the Hermitian one.

The matrix inversion algorithm can also handle ill-conditioned matrices by inverting only the part of $|b\rangle$ which is in the well-conditioned part of the matrix. Formally, instead of transforming $|b\rangle = \sum_j \beta_j |u_j\rangle$ to $|x\rangle = \sum_j \lambda_j^{-1} \beta_j |u_j\rangle$, we

transform it to a state which is close to

$$\sum_{j,\lambda_j<1/\kappa} \lambda_j^{-1}\beta_j |u_j\rangle |\text{well}\rangle + \sum_{j,\lambda_j\geq 1/\kappa} \beta_j |u_j\rangle |\text{ill}\rangle$$

in time proportional to κ^2 for any chosen κ (i.e. not necessarily the true condition number of A). The last qubit is a flag which enables the user to estimate what the size of the ill-conditioned part, or to handle it in any other way she wants. This behavior can be advantageous if we know that A is not invertible and we are interested in the projection of $|b\rangle$ on the well conditioned part of A.

Another method that is often used in classical algorithms to handle ill-conditioned matrices is to apply a preconditioner[24]. If we have a method of generating a preconditioner matrix B such that $\kappa(AB)$ is smaller than $\kappa(A)$, then we can solve $A\vec{x} = \vec{b}$ by instead solving $(AB)\vec{c} = B\vec{b}$, a matrix inversion problem with a better-conditioned matrix. Further, if A and B are both sparse, then AB is as well. Thus, as long as a state proportional to $B|b\rangle$ can be efficiently prepared, our algorithm could potentially run much faster if a suitable preconditioner is used.

The outputs of the algorithm can also be generalized. We can estimate degree-$2k$ polynomials in the entries of $\vec{x}$ by generating k copies of $|x\rangle$ and measuring the nk-qubit observable

$$\sum_{i_1,\ldots,i_k,j_1,\ldots,j_k} M_{i_1,\ldots,i_k,j_1,\ldots,j_k} |i_1,\ldots,i_k\rangle \langle j_1,\ldots,j_k|$$

on the state $|x\rangle^{\otimes k}$. Alternatively, one can use our algorithm to generate a quantum analogue of Monte-Carlo, where given A and $\vec{b}$ we sample from the vector $\vec{x}$, meaning that the value i occurs with probability $|\vec{x}_i|^2$.

Perhaps the most far-reaching generalization of the matrix inversion algorithm is not to invert matrices at all! Instead, it can compute $f(A)|b\rangle$ for any computable f. Depending on the degree of nonlinearity of f, nontrivial tradeoffs between accuracy and efficiency arise. Some variants of this idea are considered in [25, 14].

Finally, we consider physical aspects regarding the implementation and performance of the quantum matrix inversion algorithm. The algorithm allows considerable flexibility in how the matrix to be inverted A, the measurement matrix M, and the vector $\vec{b}$ are represented. If the entries of A, M represent data, they can be stored in quantum memory: a large quantum memory ($O(Ns)$ slots) is needed, but the physical requirements for quantum memory are significantly less demanding than those for full-blown quantum computation [26]. Alternatively, if A or M represents some computable transformation, e.g., a discretized differential operator, then its entries can be computed in quantum parallel, in which case no quantum memory is required. Similarly, the components of $\vec{b}$ could either be computed or stored in quantum memory in a form that allows the construction of $|b\rangle$, e.g. using Ref. [16]. No matter how the inputs are represented, the quantum processor that performs the algorithm itself

is exponentially smaller than the matrix to be inverted: a quantum computer with under one hundred qubits suffices to invert a matrix with Avogadro's number of entries. Similarly, the exponential speedup of the algorithm allows it to be performed with a relatively small number of quantum operations, thereby reducing the overhead required for quantum error correction.

Acknowledgements. We thank the W.M. Keck foundation for support, and AWH thanks them as well as MIT for hospitality while this work was carried out. AWH was also funded by the U.K. EPSRC grant "QIP IRC." SL thanks R. Zecchina for encouraging him to work on this problem. D. Farmer, M. Tegmark, S. Mitter, and P. Parillo supplied useful applications for this algorithm. We are grateful as well to R. Cleve, S. Gharabian and D. Spielman for helpful discussions.

References

[1] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In S. Goldwasser, editor, *Proceedings: 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society Press, 1994.

[2] S. Lloyd. Universal quantum simulators. *Science*, 273:1073–1078, August 1996.

[3] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for solving linear systems of equations, 2009. Supplementary material.

[4] D.G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, and Applications*. Wiley, New York, 1979.

[5] H. Buhrman, R. Cleve, J. Watrous, and R. De Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16):167902–167902, 2001.

[6] Paul Valiant. Testing symmetric properties of distributions. In *STOC*, pages 383–392, 2008.

[7] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem. *Science*, 292(5516):472–475, 2001.

[8] E. Knill, R. Laflamme, and GJ Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409:46–52, 2001.

[9] D. Aharanov and I. Arad. The BQP-hardness of approximating the Jones Polynomial, 2006. arXiv:quant-ph/0605181.

[10] M.A. Nielsen, M.R. Dowling, M. Gu, and A.C. Doherty. Quantum Computation as Geometry, 2006.

[11] M.H. Freedman, M. Larsen, and Z. Wang. A modular functor which is universal for quantum computation. *Comm. Math. Phys.*, 227(3):605–622, 2002.

[12] M.H. Freedman, A. Kitaev, M.J. Larsen, and Z. Wang. Topological quantum computation. *Bull. Am. Math. Soc.*, 40(1):31–38, 2003.

[13] A. Klappenecker and M. Roetteler. Quantum Physics Title: Engineering Functional Quantum Algorithms. *Phys. Rev. A*, 67:010302, 2003.

[14] S. K. Leyton and T. J. Osborne. A quantum algorithm to solve nonlinear differential equations, 2008. arXiv:0812.4423.

[15] D.W. Berry, G. Ahokas, R. Cleve, and B.C. Sanders. Efficient Quantum Algorithms for Simulating Sparse Hamiltonians. *Comm. Math. Phys.*, 270(2):359–371, 2007. arXiv:quant-ph/0508139.

[16] L. Grover and T. Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. arXiv:quant-ph/0208112.

[17] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum Algorithms Revisited, 1997. arXiv:quant-ph/9708016.

[18] V. Buzek, R. Derka, and S. Massar. Optimal quantum clocks. *Phys. Rev. Lett.*, 82:2207–2210, 1999. arXiv:quant-ph/9808042.

[19] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. *Quantum Amplitude Amplification and Estimation*, volume 305 of *Contemporary Mathematics Series Millenium Volume*. AMS, 2002. arXiv:quant-ph/0005055.

[20] Jonathan R. Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, March 1994.

[21] Daniel R. Simon. On the power of quantum computation. *SIAM J. Comp.*, 26:116–123, 1997.

[22] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. A limit on the speed of quantum computation in determining parity. *Phys. Rev. Lett.*, 81:5442–5444, 1998. arXiv:quant-ph/9802045.

[23] Andrew M. Childs. On the relationship between continuous- and discrete-time quantum walk, 2008. arXiv:0810.0312.

[24] K. Chen. *Matrix preconditioning techniques and applications*. Cambridge Univ. Press, Cambridge, U.K., 2005.

[25] L. Sheridan, D. Maslov, and M. Mosca. Approximating fractional time quantum evolution, 2008. arXiv:0810.3843.

[26] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Phys. Rev. Lett.*, 100:160501, 2008.

[27] D. Aharonov and A. Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing (STOC)*, pages 20–29. ACM Press New York, NY, USA, 2003. arXiv:quant-ph/0301023.

[28] P.C. Hansen. *Rank-deficient and discrete ill-posted problems: Numerical aspects of linear inversion.* SIAM, Philadelphia, PA, 1998.

[29] M. Sipser. *Introduction to the Theory of Computation.* International Thomson Publishing, 1996.

[30] C.H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. The strengths and weaknesses of quantum computation. *SIAM Journal on Computing*, 26:1510–1523, 1997.

A Supplementary Online Material

In this appendix, we describe and analyze our algorithm in full detail. While the body of the paper attempted to convey the spirit of the procedure and left out various improvements, here we take the opposite approach and describe everything, albeit possibly in a less intuitive way. We also describe in more detail our reductions from non-Hermitian matrix inversion to Hermitian matrix inversion (Section A.4) and from a general quantum computation to matrix inversion (Section A.5).

As inputs we require a procedure to produce the state $|b\rangle$, a method of producing the $\leq s$ non-zero elements of any row of A and a choice of cutoff κ. Our run-time will be roughly quadratic in κ and our algorithm is guaranteed to be correct if $\|A\| \leq 1$ and $\|A^{-1}\| \leq \kappa$.

The condition number is a crucial parameter in the algorithm. Here we present one possible method of handling ill-conditioned matrices. We will define the well-conditioned part of A to be the span of the eigenspaces corresponding to eigenvalues $\geq 1/\kappa$ and the ill-conditioned part to be the rest. Our strategy will be to flag the ill-conditioned part of the matrix (without inverting it), and let the user choose how to further handle this. Since we cannot exactly resolve any eigenvalue, we can only approximately determine whether vectors are in the well- or ill-conditioned subspaces. Accordingly, we choose some $\kappa' > \kappa$ (say $\kappa' = 2\kappa$). Our algorithm then inverts the well-conditioned part of the matrix, flags any eigenvector with eigenvalue $\leq 1/\kappa'$ as ill-conditioned, and interpolates between these two behaviors when $1/\kappa' < |\lambda| < 1/\kappa$. This is described formally in the next section. We present this strategy not because it is necessarily ideal in all cases, but because it gives a concrete illustration of the key components of our algorithm.

Finally, the algorithm produces $|x\rangle$ only up to some error ϵ which is given as part of the input. We work only with pure states, and so define error in terms

of distance between vectors, i.e. $\| \, |\alpha\rangle - |\beta\rangle \, \| = \sqrt{2(1 - \mathrm{Re}\,\langle\alpha|\beta\rangle)}$. Since ancilla states are produced and then imperfectly uncomputed by the algorithm, our output state will technically have high fidelity not with $|x\rangle$ but with $|x\rangle\,|000\ldots\rangle$. In general we do not write down ancilla qubits in the $|0\rangle$ state, so we write $|x\rangle$ instead of $|x\rangle\,|000\ldots\rangle$ for the target state, $|b\rangle$ instead of $|b\rangle\,|000\ldots\rangle$ for the initial state, and so on.

A.1 Detailed description of the algorithm

To produce the input state $|b\rangle$, we assume that there exists an efficiently-implementable unitary B, which when applied to $|\text{initial}\rangle$ produces the state $|b\rangle$, possibly along with garbage in an ancilla register. We make no further assumption about B; it may represent another part of a larger algorithm, or a standard state-preparation procedure such as [16]. Let T_B be the number of gates required to implement B. We neglect the possibility that B errs in producing $|b\rangle$ since, without any other way of producing or verifying the state $|b\rangle$, we have no way to mitigate these errors. Thus, any errors in producing $|b\rangle$ necessarily translate directly into errors in the final state $|x\rangle$.

Next, we define the state

$$|\Psi_0\rangle = \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin \frac{\pi(\tau + \frac{1}{2})}{T} \, |\tau\rangle \tag{6}$$

for a T to be chosen later. Using [16], we can prepare $|\Psi_0\rangle$ up to error ϵ_Ψ in time $\mathrm{poly}\log(T/\epsilon_\Psi)$.

One other subroutine we will need is Hamiltonian simulation. Using the reductions described in Section A.4, we can assume that A is Hermitian. To simuluate e^{iAt} for some $t \geq 0$, we use the algorithm of [15]. If A is s-sparse, $t \leq t_0$ and we want to guarantee that the error is $\leq \epsilon_H$, then this requires time

$$T_H = O(\log(N)(\log^*(N))^2 s^2 t_0 9^{\sqrt{\log(s^2 t_0/\epsilon_H)}}). = \tilde{O}(\log(N) s^2 t_0) \tag{7}$$

The scaling here is better than any power of $1/\epsilon_H$, which means that the additional error introduced by this step introduces is negligible compared with the rest of the algorithm, and the runtime is almost linear with t_0. Note that this is the only step where we require that A be sparse; as there are some other types of Hamiltonians which can be simulated efficiently (e.g. [27, 15, 23]), this broadens the set of matrices we can handle.

The key subroutine of the algorithm, denoted U_{invert}, is defined as follows:

1. Prepare $|\Psi_0\rangle^C$ from $|0\rangle$ up to error ϵ_Ψ.
2. Apply the conditional Hamiltonian evolution $\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau|^C \otimes e^{iA\tau t_0/T}$ up to error ϵ_H.
3. Apply the Fourier transform to the register C. Denote the resulting basis states with $|k\rangle$, for $k = 0, \ldots T-1$. Define $\tilde{\lambda}_k := 2\pi k/t_0$.

4. Adjoin a three-dimensional register S in the state

$$\left|h(\tilde{\lambda}_k)\right\rangle^S := \sqrt{1 - f(\tilde{\lambda}_k)^2 - g(\tilde{\lambda}_k)^2}\,|\text{nothing}\rangle^S + f(\tilde{\lambda}_k)\,|\text{well}\rangle^S + g(\tilde{\lambda}_k)\,|\text{ill}\rangle^S,$$

for functions $f(\lambda), g(\lambda)$ defined below in (8). Here 'nothing' indicates that the desired matrix inversion hasn't taken place, 'well' indicates that it has, and 'ill' means that part of $|b\rangle$ is in the ill-conditioned subspace of A.

5. Reverse steps 1-3, uncomputing any garbage produced along the way.

The functions $f(\lambda), g(\lambda)$ are known as filter functions[28], and are chosen so that for some constant $C > 1$: $f(\lambda) = 1/C\kappa\lambda$ for $\lambda \geq 1/\kappa$, $g(\lambda) = 1/C$ for $\lambda \leq 1/\kappa' := 1/2\kappa$ and $f^2(\lambda) + g^2(\lambda) \leq 1$ for all λ. Additionally, $f(\lambda)$ should satisfy a certain continuity property that we will describe in the next section. Otherwise the functions are arbitrary. One possible choice is

$$f(\lambda) = \begin{cases} \frac{1}{2\kappa\lambda} & \text{when } \lambda \geq 1/\kappa \\ \frac{1}{2}\sin\left(\frac{\pi}{2} \cdot \frac{\lambda - \frac{1}{\kappa'}}{\frac{1}{\kappa} - \frac{1}{\kappa'}}\right) & \text{when } \frac{1}{\kappa} > \lambda \geq \frac{1}{\kappa'} \\ 0 & \text{when } \frac{1}{\kappa'} > \lambda \end{cases} \tag{8a}$$

$$g(\lambda) = \begin{cases} 0 & \text{when } \lambda \geq 1/\kappa \\ \frac{1}{2}\cos\left(\frac{\pi}{2} \cdot \frac{\lambda - \frac{1}{\kappa'}}{\frac{1}{\kappa} - \frac{1}{\kappa'}}\right) & \text{when } \frac{1}{\kappa} > \lambda \geq \frac{1}{\kappa'} \\ \frac{1}{2} & \text{when } \frac{1}{\kappa'} > \lambda \end{cases} \tag{8b}$$

If U_{invert} is applied to $|u_j\rangle$ it will, up to an error we will discuss below, adjoin the state $|h(\lambda_j)\rangle$. Instead if we apply U_{invert} to $|b\rangle$ (i.e. a superposition of different $|u_j\rangle$), measure S and obtain the outcome 'well', then we will have approximately applied an operator proportional to A^{-1}. Let $\tilde{p}$ (computed in the next section) denote the success probability of this measurement. Rather than repeating $1/\tilde{p}$ times, we will use amplitude amplification [19] to obtain the same results with $O(1/\sqrt{\tilde{p}})$ repetitions. To describe the procedure, we introduce two new operators:

$$R_{\text{succ}} = I^S - 2|\text{well}\rangle\langle\text{well}|^S,$$

acting only on the S register and

$$R_{\text{init}} = I - 2|\text{initial}\rangle\langle\text{initial}|.$$

Our main algorithm then follows the amplitude amplification procedure: we start with $U_{\text{invert}}B\,|\text{initial}\rangle$ and repeatedly apply $U_{\text{invert}}BR_{\text{init}}B^\dagger U_{\text{invert}}^\dagger R_{\text{succ}}$. Finally we measure S and stop when we obtain the result 'well'. The number of repetitions would ideally be $\pi/4\sqrt{\tilde{p}}$, which in the next section we will show is $O(\kappa)$. While $\tilde{p}$ is initially unknown, the procedure has a constant probability of success if the number of repetitions is a constant fraction of $\pi/4\tilde{p}$. Thus, following [19] we repeat the entire procedure with a geometrically increasing number of repetitions each time: 1, 2, 4, 8, ..., until we have reached a power

of two that is $\geq \kappa$. This yields a constant probability of success using $\leq 4\kappa$ repetitions.

Putting everything together, the run-time is $\tilde{O}(\kappa(T_B + t_0 s^2 \log(N))$, where the $\tilde{O}$ suppresses the more-slowly growing terms of $(\log^*(N))^2$, $\exp(O(1/\sqrt{\log(t_0/\epsilon_H)}))$ and $\operatorname{poly}\log(T/\epsilon_\Psi)$. In the next section, we will show that t_0 can be taken to be $O(\kappa/\epsilon)$ so that the total run-time is $\tilde{O}(\kappa T_B + \kappa^2 s^2 \log(N)/\epsilon)$.

A.2 Error Analysis

In this section we show that taking $t_0 = O(\kappa/\epsilon)$ introduces an error of $\leq \epsilon$ in the final state. The main subtlety in analyzing the error comes from the post-selection step, in which we choose only the part of the state attached to the $|\text{well}\rangle$ register. This can potentially magnify errors in the overall state. On the other hand, we may also be interested in the non-postselected state, which results from applying U_{invert} a single time to $|b\rangle$. For instance, this could be used to estimate the amount of weight of $|b\rangle$ lying in the ill-conditioned components of A. Somewhat surprisingly, we show that the error in both cases is upper-bounded by $O(\kappa/t_0)$.

In this section, it will be convenient to ignore the error terms ϵ_H and ϵ_Ψ, as these can be made negligible with relatively little effort and it is the errors from phase estimation that will dominate. Let $\tilde{U}$ denote a version of U_{invert} in which everything except the phase estimation is exact. Since $\|\tilde{U} - U_{\text{invert}}\| \leq O(\epsilon_H + \epsilon_\Psi)$, it is sufficient to work with $\tilde{U}$. Define U to be the ideal version of U_{invert} in which there is no error in any step.

Theorem A.1 (Error bounds)**.**

1. *In the case when no post-selection is performed, the error is bounded as*

$$\|\tilde{U} - U\| \leq O(\kappa/t_0). \tag{9}$$

2. *If we post-select on the flag register being in the space spanned by $\{|well\rangle, |ill\rangle\}$ and define the normalized ideal state to be $|x\rangle$ and our actual state to be $|\tilde{x}\rangle$ then*

$$\| \, |\tilde{x}\rangle - |x\rangle \, \| \leq O(\kappa/t_0). \tag{10}$$

3. *If $|b\rangle$ is entirely within the well-conditioned subspace of A and we post-select on the flag register being $|well\rangle$ then*

$$\| \, |\tilde{x}\rangle - |x\rangle \, \| \leq O(\kappa/t_0). \tag{11}$$

The third claim is often of the most practical interest, but the other two are useful if we want to work with the ill-conditioned space, or estimate its weight.

The rest of the section is devoted to the proof of Theorem A.1. We first show that the third claim is a corollary of the second, and then prove the first two claims more or less independently. To prove (10 assuming (9), observe that if $|b\rangle$ is entirely in the well-conditioned space, the ideal state $|x\rangle$ is proportional

to $A^{-1}|b\rangle|\text{well}\rangle$. Model the post-selection on $|\text{well}\rangle$ by a post-selection first on the space spanned by $\{|\text{well}\rangle, |\text{ill}\rangle\}$, followed by a post-selection onto $|\text{well}\rangle$. By (9), the first post-selection leaves us with error $O(\kappa/t_0)$. This implies that the second post-selection will succeed with probability $\geq 1-O(\kappa^2/t_0^2)$ and therefore will increase the error by at most $O(\kappa/t_0)$. The final error is then $O(\kappa/t_0)$ as claimed in (11).

Now we turn to the proof of (9). A crucial piece of the proof will be the following statement about the continuity of $|h(\lambda)\rangle$.

Lemma A.2. *The map $\lambda \mapsto |h(\lambda)\rangle$ is $O(\kappa)$-Lipschitz, meaning that for any $\lambda_1 \neq \lambda_2$,*

$$\| \, |h(\lambda_1)\rangle - |h(\lambda_2)\rangle \, \| = \sqrt{2(1 - Re\,\langle h(\lambda_1)|h(\lambda_2)\rangle)} \leq c\kappa|\lambda_1 - \lambda_2|,$$

for some $c = O(1)$.

Proof. Since $\lambda \mapsto |h(\lambda)\rangle$ is continuous everywhere and differentiable everywhere except at $1/\kappa$ and $1/\kappa'$, it suffices to bound the norm of the derivative of $|h(\lambda)\rangle$. We consider it piece by piece. When $\lambda > 1/\kappa$,

$$\frac{d}{d\lambda}|h(\lambda)\rangle = \frac{1}{2\kappa^2\lambda^3\sqrt{1-1/2\kappa^2\lambda^2}}|\text{nothing}\rangle - \frac{1}{2\kappa\lambda^2}|\text{well}\rangle,$$

which has squared norm $\frac{1}{2\kappa^2\lambda^4(2\kappa^2\lambda^2-1)} + \frac{1}{4\kappa^2\lambda^4} \leq \kappa^2$. Next, when $1/\kappa' < \lambda < 1/\kappa$, the norm of $\frac{d}{d\lambda}|h(\lambda)\rangle$ is

$$\frac{1}{2} \cdot \frac{\pi}{2} \cdot \frac{1}{\frac{1}{\kappa} - \frac{1}{\kappa'}} = \frac{\pi}{2}\kappa.$$

Finally $\frac{d}{d\lambda}|h(\lambda)\rangle = 0$ when $\lambda < 1/\kappa'$. This completes the proof, with $c = \frac{\pi}{2}$. □

Now we return to the proof of (9). Let $\tilde{P}$ denote the first three steps of the algorithm. They can be thought of as mapping the initial zero qubits to a $|k\rangle$ register, together with some garbage, as follows:

$$\tilde{P} = \sum_{j=1}^{n} |u_j\rangle\langle u_j| \otimes \sum_k \alpha_{k|j}|k\rangle|\text{garbage}(j,k)\rangle\langle\text{initial}|,$$

where the guarantee that the phase estimation algorithm gives us is that $\alpha_{k|j}$ is concentrated around $\lambda_j \approx 2\pi k/t_0 =: \tilde{\lambda}_k$. Technically, $\tilde{P}$ should be completed to make it a unitary operator by defining some arbitrary behavior on inputs other than $|\text{initial}\rangle$ in the last register.

Consider a test state $|b\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle$. The ideal functionality is defined by

$$|\varphi\rangle = U|b\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle|h(\lambda_j)\rangle,$$

while the actual algorithm produces the state

$$|\tilde{\varphi}\rangle = \tilde{U}\,|b\rangle = \tilde{P}^\dagger \sum_{j=1}^{N} \beta_j\,|u_j\rangle \sum_k \alpha_{k|j}\,|k\rangle \left|h(\tilde{\lambda}_k)\right\rangle,$$

We wish to calculate $\langle\tilde{\varphi}|\varphi\rangle$, or equivalently the inner product between $\tilde{P}\,|\tilde{\varphi}\rangle$ and $\tilde{P}\,|\varphi\rangle = \sum_{j,k} \beta_j \alpha_{k|j}\,|u_j\rangle\,|k\rangle\,|h(\lambda_j)\rangle$. This inner product is

$$\langle\tilde{\varphi}|\varphi\rangle = \sum_{j=1}^{N} |\beta_j|^2 \sum_k |\alpha_{k|j}|^2 \left\langle h(\tilde{\lambda}_k)|h(\lambda_j)\right\rangle := \mathbb{E}_j \mathbb{E}_k \left\langle h(\tilde{\lambda}_k)|h(\lambda_j)\right\rangle,$$

where we think of j and k as random variables with joint distribution $\Pr(j,k) = |\beta_j|^2|\alpha_{k|j}|^2$. Thus

$$\mathrm{Re}\,\langle\tilde{\varphi}|\varphi\rangle = \mathbb{E}_j \mathbb{E}_k \mathrm{Re} \left\langle h(\tilde{\lambda}_k)|h(\lambda_j)\right\rangle.$$

Let $\delta = \lambda_j t_0 - 2\pi k = t_0(\lambda_j - \tilde{\lambda}_k)$. From Lemma A.2, $\mathrm{Re}\left\langle h(\tilde{\lambda}_k)|h(\lambda_j)\right\rangle \geq 1 - c^2\kappa^2\delta^2/2t_0^2$, where $c \leq \frac{\pi}{2}$ is a constant. There are two sources of infidelity. For $\delta \leq 2\pi$, the inner product is at least $1 - 2\pi^2c^2\kappa^2/t_0^2$. For larger values of δ, we use the bound $|\alpha_{k|j}|^2 \leq 64\pi^2/(\lambda_j t_0 - 2\pi k)^4$ (proved in Section A.3) to find an infidelity contribution that is

$$\leq 2 \sum_{k=\frac{\lambda_j t_0}{2\pi}+1}^{\infty} \frac{64\pi^2}{\delta^4} \frac{c^2\kappa^2\delta^2}{2t_0^2} = \frac{64\pi^2c^2\kappa^2}{t_0^2} \sum_{k=1}^{\infty} \frac{1}{4\pi^2k^2} = \frac{8\pi^2c^2}{3} \cdot \frac{\kappa^2}{t_0^2}.$$

Summarizing, we find that $\mathrm{Re}\,\langle\tilde{\varphi}|\varphi\rangle \geq 1 - 5\pi^2c^2\kappa^2/t_0^2$, which translates into $\| \,|\tilde{\varphi}\rangle - |\varphi\rangle\, \| \leq 4\pi c\kappa/t_0 = 2\pi^2\kappa/t_0$. Since the initial state $|b\rangle$ was arbitrary, this bounds the operator distance $\|\tilde{U} - U\|$ as claimed in (9).

Turning now to the post-selected case, we observe that

$$|x\rangle := \frac{f(A)\,|b\rangle\,|\mathrm{well}\rangle + g(A)\,|b\rangle\,|\mathrm{ill}\rangle}{\sqrt{\langle b|\,(f(A)^2 + g(A)^2)\,|b\rangle}} \tag{12}$$

$$= \frac{\sum_j \beta_j\,|u_j\rangle\,(f(\lambda_j)\,|\mathrm{well}\rangle + g(\lambda_j)\,|\mathrm{ill}\rangle)}{\sqrt{\sum_j |\beta_j|^2(f(\lambda_j)^2 + g(\lambda_j)^2)}} \tag{13}$$

$$=: \frac{\sum_j \beta_j\,|u_j\rangle\,(f(\lambda_j)\,|\mathrm{well}\rangle + g(\lambda_j)\,|\mathrm{ill}\rangle)}{\sqrt{p}}. \tag{14}$$

Where in the last step we have defined

$$p := \mathbb{E}_j[f(\lambda_j)^2 + g(\lambda_j)^2]$$

to be the probability that the post-selection succeeds. Naively, this post-selection could magnify the errors by as much as $1/\sqrt{p}$, but by careful examination of the errors, we find that this worst-case situation only occurs when

the errors are small in the first place. This is what will allow us to obtain the same $O(\kappa/t_0)$ error bound even in the post-selected state.

Now write the actual state that we produce as

$$|\tilde{x}\rangle := \frac{\tilde{P}^\dagger \sum_{j=1}^N \beta_j |u_j\rangle \sum_k \alpha_{k|j} |k\rangle \left(f(\tilde{\lambda}_k)|\text{well}\rangle + g(\tilde{\lambda}_k)|\text{ill}\rangle\right)}{\sqrt{\mathbb{E}_{j,k} f(\tilde{\lambda}_k)^2 + g(\tilde{\lambda}_k)^2}} \tag{15}$$

$$=: \frac{\tilde{P}^\dagger \sum_{j=1}^N \beta_j |u_j\rangle \sum_k \alpha_{k|j} |k\rangle \left(f(\tilde{\lambda}_k)|\text{well}\rangle + g(\tilde{\lambda}_k)|\text{ill}\rangle\right)}{\sqrt{\tilde{p}}}, \tag{16}$$

where we have defined $\tilde{p} = \mathbb{E}_{j,k}[f(\tilde{\lambda}_k)^2 + g(\tilde{\lambda}_k)^2]$.

Recall that j and k are random variables with joint distribution $\Pr(j,k) = |\beta_j|^2 |\alpha_{k|j}|^2$. We evaluate the contribution of a single j value. Define $\lambda := \lambda_j$ and $\tilde{\lambda} := 2\pi k/t_0$. Note that $\delta = t_0(\lambda - \tilde{\lambda})$ and that $\mathbb{E}\delta, \mathbb{E}\delta^2 = O(1)$. Here δ depends implicitly on both j and k, and the above bounds on its expectations hold even when conditioning on an arbitrary value of j. We further abbreviate $f := f(\lambda)$, $\tilde{f} := \tilde{f}(\lambda)$, $g := g(\lambda)$ and $\tilde{g} = \tilde{g}(\lambda)$. Thus $p := \mathbb{E}[f^2 + g^2]$ and $\tilde{p} = \mathbb{E}[\tilde{f}^2 + \tilde{g}^2]$.

Our goal is to bound $\| \, |x\rangle - |\tilde{x}\rangle \, \|$ in (10). We work instead with the fidelity

$$F := \langle \tilde{x}|x\rangle = \frac{\mathbb{E}[f\tilde{f} + g\tilde{g}]}{\sqrt{p\tilde{p}}} = \frac{\mathbb{E}[f^2 + g^2] + \mathbb{E}[(\tilde{f} - f)f + (\tilde{g} - g)g]}{p\sqrt{1 + \frac{\tilde{p} - p}{p}}} \tag{17}$$

$$= \frac{1 + \frac{\mathbb{E}[(\tilde{f} - f)f + (\tilde{g} - g)g]}{p}}{\sqrt{1 + \frac{\tilde{p} - p}{p}}} \geq \left(1 + \frac{\mathbb{E}[(\tilde{f} - f)f + (\tilde{g} - g)g]}{p}\right)\left(1 - \frac{1}{2} \cdot \frac{\tilde{p} - p}{p}\right) \tag{18}$$

Next we expand

$$\tilde{p} - p = \mathbb{E}[\tilde{f}^2 - f^2] + \mathbb{E}[\tilde{g}^2 - g^2] \tag{19}$$

$$= \mathbb{E}[(\tilde{f} - f)(\tilde{f} + f)] + \mathbb{E}[(\tilde{g} - g)(\tilde{g} + g)] \tag{20}$$

$$= 2\mathbb{E}[(\tilde{f} - f)f] + 2\mathbb{E}[(\tilde{g} - g)g] + \mathbb{E}[(\tilde{f} - f)^2] + \mathbb{E}[(\tilde{g} - g)^2] \tag{21}$$

Substituting into (18), we find

$$F \geq 1 - \frac{\mathbb{E}[(\tilde{f} - f)^2 + (\tilde{g} - g)^2]}{2p} - \frac{\mathbb{E}[(\tilde{f} - f)f + (\tilde{g} - g)g]}{p} \cdot \frac{\tilde{p} - p}{2p} \tag{22}$$

We now need an analogue of the Lipschitz condition given in Lemma A.2.

Lemma A.3. *Let $f, \tilde{f}, g, \tilde{g}$ be defined as above, with $\kappa' = 2\kappa$. Then*

$$|f - \tilde{f}|^2 + |g - \tilde{g}|^2 \leq c\frac{\kappa^2}{t_0^2}\delta^2 |f^2 + g^2|$$

where $c = \pi^2/2$.

Proof. Remember that $\tilde{f} = f(\lambda - \delta/t_0)$ and similarly for $\tilde{g}$.

Consider the case first when $\lambda \geq 1/\kappa$. In this case $g = 0$, and we need to show that

$$|f - \tilde{f}| \leq 2\frac{\kappa|\delta f|}{t_0} = \frac{|\lambda - \tilde{\lambda}|}{\lambda} \tag{23}$$

To prove this, we consider four cases. First, suppose $\tilde{\lambda} \geq 1/\kappa$. Then $|f - \tilde{f}| = \frac{1}{2\kappa}\frac{|\tilde{\lambda}-\lambda|}{\tilde{\lambda}\cdot\lambda} \leq |\delta|/2t_0\lambda$. Next, suppose $\lambda = 1/\kappa$ (so $f = 1/2$) and $\tilde{\lambda} < 1/\kappa$. Since $\sin\frac{\pi}{2}\alpha \geq \alpha$ for $0 \leq \alpha \leq 1$, we have

$$|f - \tilde{f}| \leq \frac{1}{2} - \frac{1}{2}\frac{\tilde{\lambda} - \frac{1}{\kappa'}}{\frac{1}{\kappa} - \frac{1}{\kappa'}} = \frac{1}{2} - \kappa(\tilde{\lambda} - \frac{1}{2}) = \kappa(\frac{1}{\kappa} - \tilde{\lambda}), \tag{24}$$

and using $\lambda = 1/\kappa$ we find that $|f - \tilde{f}| = \frac{\lambda-\tilde{\lambda}}{\lambda}$, as desired. Next, if $\tilde{\lambda} < 1/\kappa < \lambda$ and $f < \tilde{f}$ then replacing λ with $1/\kappa$ only makes the inequality tighter. Finally, suppose $\tilde{\lambda} < 1/\kappa < \lambda$ and $\tilde{f} < f$. Using (24) and $\lambda > 1/\kappa$ we find that $f - \tilde{f} \leq 1 - \kappa\tilde{\lambda} < 1 - \tilde{\lambda}/\lambda = (\lambda - \tilde{\lambda})/\lambda$, as desired.

Now, suppose that $\lambda < 1/\kappa$. Then

$$|f - \tilde{f}|^2 \leq \frac{\delta^2}{t_0^2}\max|f'|^2 = \frac{\pi^2}{4}\frac{\delta^2}{t_0^2}\kappa^2.$$

And similarly

$$|g - \tilde{g}|^2 \leq \frac{\delta^2}{t_0^2}\max|g'|^2 = \frac{\pi^2}{4}\frac{\delta^2}{t_0^2}\kappa^2.$$

Finally $f(\lambda)^2 + g(\lambda)^2 = 1/2$ for any $\lambda \leq 1/\kappa$, implying the result. □

Now we use Lemma A.3 to bound the two error contributions in (18). First bound

$$\frac{\mathbb{E}[(\tilde{f} - f)^2 + (\tilde{g} - g)^2]}{2p} \leq O\left(\frac{\kappa^2}{t_0^2}\right) \cdot \frac{\mathbb{E}[(f^2 + g^2)\delta^2]}{\mathbb{E}[f^2 + g^2]} \leq O\left(\frac{\kappa^2}{t_0^2}\right) \tag{25}$$

The first inequality used Lemma A.3 and the second used the fact that $\mathbb{E}[\delta^2] \leq O(1)$ even when conditioned on an arbitrary value of j (or equivalently λ_j).

Next,

$$\frac{\mathbb{E}[(\tilde{f} - f)f + (\tilde{g} - g)g]}{p} \leq \frac{\mathbb{E}\left[\sqrt{\left((\tilde{f} - f)^2 + (\tilde{g} - g)^2\right)(f^2 + g^2)}\right]}{p} \tag{26}$$

$$\leq \frac{\mathbb{E}\left[\sqrt{\frac{\delta^2\kappa^2}{t_0^2}(f^2 + g^2)^2}\right]}{p} \tag{27}$$

$$\leq O\left(\frac{\kappa}{t_0}\right), \tag{28}$$

where the first inequality is Cauchy-Schwartz, the second is Lemma A.3 and the last uses the fact that $\mathbb{E}[|\delta|] \leq \sqrt{\mathbb{E}[\delta^2]} = O(1)$ even when conditioned on j.

We now substitute (25) and (28) into (21) (and assume $\kappa \leq t_0$) to find

$$\frac{|\tilde{p} - p|}{p} \leq O\left(\frac{\kappa}{t_0}\right). \tag{29}$$

Substituting (25), (28) and (29) into (22), we find $\mathrm{Re}\,\langle \tilde{x}|x\rangle \geq 1 - O(\kappa^2/t_0^2)$, or equivalently, that $\| \, |\tilde{x}\rangle - |x\rangle \, \| \leq \epsilon$. This completes the proof of Theorem A.1. □

A.3 Phase estimation calculations

Here we describe, in our notation, the improved phase-estimation procedure of [18], and prove the concentration bounds on $|\alpha_{k|j}|$. Adjoin the state

$$|\Psi_0\rangle = \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin \frac{\pi(\tau + \frac{1}{2})}{T} \, |\tau\rangle \, .$$

Apply the conditional Hamiltonian evolution $\sum_\tau |\tau\rangle\langle\tau| \otimes e^{iA\tau t_0/T}$. Assume the target state is $|u_j\rangle$, so this becomes simply the conditional phase $\sum_\tau |\tau\rangle\langle\tau| e^{i\lambda_j t_0 \tau/T}$. The resulting state is

$$\left|\Psi_{\lambda_j t_0}\right\rangle = \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} e^{\frac{i\lambda_j t_0 \tau}{T}} \sin \frac{\pi(\tau + \frac{1}{2})}{T} \, |\tau\rangle \, |u_j\rangle \, .$$

We now measure in the Fourier basis, and find that the inner product with $\frac{1}{\sqrt{T}}\sum_{\tau=0}^{T-1} e^{\frac{2\pi i k\tau}{T}} |\tau\rangle |u_j\rangle$ is (defining $\delta := \lambda_j t_0 - 2\pi k$):

$$\alpha_{k|j} = \frac{\sqrt{2}}{T}\sum_{\tau=0}^{T-1} e^{i\frac{\tau}{T}(\lambda_j t_0 - 2\pi k)} \sin\frac{\pi(\tau+\frac{1}{2})}{T} \tag{30}$$

$$= \frac{1}{i\sqrt{2}T}\sum_{\tau=0}^{T-1} e^{i\frac{\tau\delta}{T}}\left(e^{\frac{i\pi(\tau+1/2)}{T}} - e^{-\frac{i\pi(\tau+1/2)}{T}}\right) \tag{31}$$

$$= \frac{1}{i\sqrt{2}T}\sum_{\tau=0}^{T-1} e^{\frac{i\pi}{2T}} e^{i\tau\frac{\delta+\pi}{T}} - e^{-\frac{i\pi}{2T}} e^{i\tau\frac{\delta-\pi}{T}} \tag{32}$$

$$= \frac{1}{i\sqrt{2}T}\left(e^{\frac{i\pi}{2T}}\frac{1-e^{i\pi+i\delta}}{1-e^{i\frac{\delta+\pi}{T}}} - e^{-\frac{i\pi}{2T}}\frac{1-e^{i\pi+i\delta}}{1-e^{i\frac{\delta-\pi}{T}}}\right) \tag{33}$$

$$= \frac{1+e^{i\delta}}{i\sqrt{2}T}\left(\frac{e^{-i\delta/2T}}{e^{-\frac{i}{2T}(\delta+\pi)} - e^{\frac{i}{2T}(\delta+\pi)}} - \frac{e^{-i\delta/2T}}{e^{-\frac{i}{2T}(\delta-\pi)} - e^{\frac{i}{2T}(\delta-\pi)}}\right) \tag{34}$$

$$= \frac{(1+e^{i\delta})e^{-i\delta/2T}}{i\sqrt{2}T}\left(\frac{1}{-2i\sin\left(\frac{\delta+\pi}{2T}\right)} - \frac{1}{-2i\sin\left(\frac{\delta-\pi}{2T}\right)}\right) \tag{35}$$

$$= -e^{i\frac{\delta}{2}(1-\frac{1}{T})}\frac{\sqrt{2}\cos(\frac{\delta}{2})}{T}\left(\frac{1}{\sin\left(\frac{\delta+\pi}{2T}\right)} - \frac{1}{\sin\left(\frac{\delta-\pi}{2T}\right)}\right) \tag{36}$$

$$= -e^{i\frac{\delta}{2}(1-\frac{1}{T})}\frac{\sqrt{2}\cos(\frac{\delta}{2})}{T}\cdot\frac{\sin\left(\frac{\delta-\pi}{2T}\right) - \sin\left(\frac{\delta+\pi}{2T}\right)}{\sin\left(\frac{\delta+\pi}{2T}\right)\sin\left(\frac{\delta-\pi}{2T}\right)} \tag{37}$$

$$= e^{i\frac{\delta}{2}(1-\frac{1}{T})}\frac{\sqrt{2}\cos(\frac{\delta}{2})}{T}\cdot\frac{2\cos\left(\frac{\delta}{2T}\right)\sin\left(\frac{\pi}{2T}\right)}{\sin\left(\frac{\delta+\pi}{2T}\right)\sin\left(\frac{\delta-\pi}{2T}\right)} \tag{38}$$

Following [18], we make the assumption that $2\pi \leq \delta \leq T/10$. Further using $\alpha - \alpha^3/6 \leq \sin\alpha \leq \alpha$ and ignoring phases we find that

$$|\alpha_{k|j}| \leq \frac{4\pi\sqrt{2}}{(\delta^2-\pi^2)(1-\frac{\delta^2+\pi^2}{3T^2})} \leq \frac{8\pi}{\delta^2}. \tag{39}$$

Thus $|\alpha_{k|j}|^2 \leq 64\pi^2/\delta^2$ whenever $|k - \lambda_j t_0/2\pi| \geq 1$.

A.4 The non-Hermitian case

Suppose $A \in \mathbb{C}^{M\times N}$ with $M \leq N$. Generically $Ax = b$ is now underconstrained. Let the singular value decomposition of A be

$$A = \sum_{j=1}^{M} \lambda_j |u_j\rangle \langle v_j| ,$$

with $|u_j\rangle \in \mathbb{C}^M$, $|v_j\rangle \in \mathbb{C}^N$ and $\lambda_1 \geq \cdots \lambda_M \geq 0$. Let $V = \mathrm{span}\{|v_1\rangle, \ldots, |v_M\rangle\}$. Define

$$H = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}. \tag{40}$$

H is Hermitian with eigenvalues $\pm\lambda_1, \ldots, \pm\lambda_M$, corresponding to eigenvectors $\left|w_j^{\pm}\right\rangle := \frac{1}{\sqrt{2}}(|0\rangle |u_j\rangle \pm |1\rangle |v_j\rangle)$. It also has $N-M$ zero eigenvalues, corresponding to the orthogonal complement of V.

To run our algorithm we use the input $|0\rangle |b\rangle$. If $|b\rangle = \sum_{j=1}^{M} \beta_j |u_j\rangle$ then

$$|0\rangle |b\rangle = \sum_{j=1}^{M} \beta_j \frac{1}{\sqrt{2}}(\left|w_j^{+}\right\rangle + \left|w_j^{-}\right\rangle)$$

and running the inversion algorithm yields a state proportional to

$$H^{-1} |0\rangle |b\rangle = \sum_{j=1}^{M} \beta_j \lambda_j^{-1} \frac{1}{\sqrt{2}}(\left|w_j^{+}\right\rangle - \left|w_j^{-}\right\rangle) = \sum_{j=1}^{M} \beta_j \lambda_j^{-1} |1\rangle |v_j\rangle .$$

Dropping the inital $|1\rangle$, this defines our solution $|x\rangle$. Note that our algorithm does not produce any component in $V^{\perp}$, although doing so would have also yielded valid solutions. In this sense, it could be said to be finding the $|x\rangle$ that minimizes $\langle x|x\rangle$ while solving $A |x\rangle = |b\rangle$.

On the other hand, if $M \geq N$ then the problem is overconstrained. Let $U = \text{span}\{|u_1\rangle, \ldots, |u_N\rangle\}$. The equation $A |x\rangle = |b\rangle$ is satisfiable only if $|b\rangle \in U$. In this case, applying H to $|0\rangle |b\rangle$ will return a valid solution. But if $|b\rangle$ has some weight in $U^{\perp}$, then $|0\rangle |b\rangle$ will have some weight in the zero eigenspace of H, which will be flagged as ill-conditioned by our algorithm. We might choose to ignore this part, in which case the algorithm will return an $|x\rangle$ satisfying $A |x\rangle = \sum_{j=1}^{N} |u_j\rangle\langle u_j| \, |b\rangle$.

A.5 Optimality

In this section, we explain in detail two important ways in which our algorithm is optimal up to polynomial factors. First, no classical algorithm can perform the same matrix inversion task; and second, our dependence on condition number and accuracy cannot be substantially improved.

We present two versions of our lower bounds; one based on complexity theory, and one based on oracles. We say that an algorithm solves matrix inversion if its input and output are

1. Input: An $O(1)$-sparse matrix A specified either via an oracle or via a poly(log(N))-time algorithm that returns the nonzero elements in a row.

2. Output: A bit that equals one with probability $\langle x| M |x\rangle \pm \epsilon$, where $M = |0\rangle\langle 0| \otimes I_{N/2}$ corresponds to measuring the first qubit and $|x\rangle$ is a normalized state proportional to $A^{-1} |b\rangle$ for $|b\rangle = |0\rangle$.

Further we demand that A is Hermitian and $\kappa^{-1} I \leq A \leq I$. We take ϵ to be a fixed constant, such as 1/100, and later deal with the dependency in ϵ. If the algorithm works when A is specified by an oracle, we say that it is relativizing. Even though this is a very weak definition of inverting matrices, this task is still hard for classical computers.

Theorem A.4. *1. If a quantum algorithm exists for matrix inversion running in time* $\kappa^{1-\delta} \cdot \mathrm{poly}\log(N)$ *for some* $\delta > 0$*, then* **BQP**=**PSPACE**.

2. No relativizing quantum algorithm can run in time $\kappa^{1-\delta} \cdot \mathrm{poly}\log(N)$.

3. If a classical algorithm exists for matrix inversion running in time $\mathrm{poly}(\kappa, \log(N))$*, then* **BPP**=**BQP**.

Given an n-qubit T-gate quantum computation, define U as in (4). Define

$$A = \begin{pmatrix} 0 & I - Ue^{-\frac{1}{T}} \\ I - U^\dagger e^{-\frac{1}{T}} & 0 \end{pmatrix}. \tag{41}$$

Note that A is Hermitian, has condition number $\kappa \leq 2T$ and dimension $N = 6T2^n$. Solving the matrix inversion problem corresponding to A produces an ϵ-approximation of the quantum computation corresponding to applying $U_1, \ldots, U_T$, assuming we are allowed to make any two outcome measurement on the output state $|x\rangle$. Recall that

$$\left(I - Ue^{-\frac{1}{T}}\right)^{-1} = \sum_{k \geq 0} U^k e^{-k/T}. \tag{42}$$

We define a measurement M_0, which outputs zero if the time register t is between $T+1$ and $2T$, and the original measurement's output was one. As $\Pr(T+1 \leq k \leq 2T) = e^{-2}/(1 + e^{-2} + e^{-4})$ and is independent of the result of the measurement M, we can estimate the expectation of M with accuracy ϵ by iterating this procedure $O\left(1/\epsilon^2\right)$ times.

In order to perform the simulation when measuring only the first qubit, define

$$B = \begin{pmatrix} I_{6T2^n} & 0 \\ 0 & I_{3T2^n} - Ue^{-\frac{1}{T}} \end{pmatrix}. \tag{43}$$

We now define $\tilde{B}$ to be the matrix B, after we permuted the rows and columns such that if

$$C = \begin{pmatrix} 0 & \tilde{B} \\ \tilde{B}^\dagger & 0 \end{pmatrix}. \tag{44}$$

and $C\vec{y} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}$, then measuring the first qubit of $|y\rangle$ would correspond to perform M_0 on $|x\rangle$. The condition number of C is equal to that of A, but the dimension is now $N = 18T2^n$.

Now suppose we could solve matrix inversion in time $\kappa^{1-\delta}(\log(N)/\epsilon)^{c_1}$ for constants $c_1 \geq 2, \delta > 0$. Given a computation with $T \leq 2^{2n}/18$, let $m = \frac{2}{\delta}\frac{\log(2n)}{\log(\log(n))}$ and $\epsilon = 1/100m$. For sufficiently large n, $\epsilon \geq 1/\log(n)$. Then

$$\kappa^{1-\delta}\left(\frac{\log(N)}{\epsilon}\right)^{c_1} \leq (2T)^{1-\delta}\left(\frac{3n}{\epsilon}\right)^{c_1} \leq T^{1-\delta}c_2(n\log(n))^{c_1},$$

where $c_2 = 2^{1-\delta}3^{c_1}$ is another constant.

We now have a recipe for simulating an n_i-qubit T_i-gate computation with $n_{i+1} = n_i + \log(18T_i)$ qubits, $T_{i+1} = T_i^{1-\delta} c_3 (n_i \log(n_i))^{c_1}$ gates and error ϵ. Our strategy is to start with an n_0-qubit T_0-gate computation and iterate this simulation $\ell \leq m$ times, ending with an n_ℓ-qubit T_ℓ-gate computation with error $\leq m\epsilon \leq 1/100$. We stop iterating either after m steps, or whenever $T_{i+1} > T_i^{1-\delta/2}$, whichever comes first. In the latter case, we set ℓ equal to the first i for which $T_{i+1} > T_i^{1-\delta/2}$.

In the case where we iterated the reduction m times, we have $T_i \leq T^{(1-\delta/2)^i} \leq 2^{(1-\delta/2)^i 2n_0}$, implying that $T_m \leq n_0$. On the other hand, suppose we stop for some $\ell < m$. For each $i < \ell$ we have $T_{i+1} \leq T_i^{1-\delta/2}$. Thus $T_i \leq 2^{(1-\delta/2)^i 2n_0}$ for each $i \leq \ell$. This allows us to bound $n_i = n_0 + \sum_{j=0}^{i-1} \log(18T_i) = n_0 + 2n_0 \sum_{j=0}^{i-1} (1-\delta/2)^j + i\log(18) \leq \left(\frac{4}{\delta} + 1\right) n_0 + m\log(18)$. Defining yet another constant, this implies that $T_{i+1} \leq T_i^{1-\delta} c_3 (n_0 \log(n_0))^{c_1}$. Combining this with our stopping condition $T_{\ell+1} > T_\ell^{1-\delta/2}$ we find that

$$T_\ell \leq (c_3 (n_0 \log(n_0))^{c_1})^{\frac{2}{\delta}} = \text{poly}(n_0).$$

Therefore, the runtime of the procedure is polynomial in n_0 regardless of the reason we stopped iterating the procedure. The number of qubits used increases only linearly.

Recall that the TQBF (totally quantified Boolean formula satisfiability) problem is **PSPACE**-complete, meaning that any k-bit problem instance for any language in **PSPACE** can be reduced to a TQBF problem of length $n = \text{poly}(k)$ (see [29] for more information). The formula can be solved in time $T \leq 2^{2n}/18$, by exhaustive enumeration over the variables. Thus a **PSPACE** computation can be solved in quantum polynomial time. This proves the first part of the theorem.

To incorporate oracles, note that our construction of U in (4) could simply replace some of the U_i's with oracle queries. This preserves sparsity, although we need the rows of A to now be specified by oracle queries. We can now iterate the speedup in exactly the same manner. However, we conclude with the ability to solve the OR problem on 2^n inputs in $\text{poly}(n)$ time and queries. This, of course, is impossible [30], and so the purported relativizing quantum algorithm must also be impossible.

The proof of part 3 of Theorem A.4 simply formulates a $\text{poly}(n)$-time, n-qubit quantum computation as a $\kappa = \text{poly}(n)$, $N = 2^n \cdot \text{poly}(n)$ matrix inversion problem and applies the classical algorithm which we have assumed exists. □

Theorem A.4 established the universality of the matrix inversion algorithm. To extend the simulation to problems which are not decision problems, note that the algorithm actually supplies us with $|x\rangle$ (up to some accuracy). For example, instead of measuring an observable M, we can measure $|x\rangle$ in the computational basis, obtaining the result i with probability $|\langle i|x\rangle|^2$. This gives a way to simulate quantum computation by classical matrix inversion algorithms. In turn, this can be used to prove lower bounds on classical matrix inversion algorithms,

where we assume that the classical algorithms output samples according to this distribution.

Theorem A.5. *No relativizing classical matrix inversion algorithm can run in time $N^\alpha 2^{\beta\kappa}$ unless $3\alpha + 4\beta \geq 1/2$.*

If we consider matrix inversion algorithms that work only on positive definite matrices, then the $N^\alpha 2^{\beta\kappa}$ bound becomes $N^\alpha 2^{\beta\sqrt{\kappa}}$.

Proof. Recall Simon's problem [21], in which we are given $f : \mathbb{Z}_2^n \to \{0,1\}^{2n}$ such that $f(x) = f(y)$ iff $x + y = a$ for some $a \in \mathbb{Z}_2^n$ that we would like to find. It can be solved by running a $3n$-qubit $2n+1$-gate quantum computation $O(n)$ times and performing a $\text{poly}(n)$ classical computation. The randomized classical lower bound is $\Omega(2^{n/2})$ from birthday arguments.

Converting Simon's algorithm to a matrix A yields $\kappa \approx 4n$ and $N \approx 36n2^{3n}$. The run-time is $N^\alpha 2^{\beta\kappa} \approx 2^{(3\alpha+4\beta)n} \cdot \text{poly}(n)$. To avoid violating the oracle lower bound, we must have $3\alpha + 4\beta \geq 1/2$, as required. □

Next, we argue that the accuracy of algorithm cannot be substantially improved. Returning now to the problem of estimating $\langle x| M |x\rangle$, we recall that classical algorithms can approximate this to accuracy ϵ in time $O(N\kappa \,\text{poly}(\log(1/\epsilon)))$. This $\text{poly}(\log(1/\epsilon))$ dependence is because when writing the vectors $|b\rangle$ and $|x\rangle$ as bit strings means that adding an additional bit will double the accuracy. However, sampling-based algorithms such as ours cannot hope for a better than $\text{poly}(1/\epsilon)$ dependence of the run-time on the error. Thus proving that our algorithm's error performance cannot be improved will require a slight redefinition of the problem.

Define the matrix inversion estimation problem as follows. Given $A, b, M, \epsilon, \kappa, s$ with $\|A\| \leq 1, \|A^{-1}\| \leq \kappa$, A s-sparse and efficiently row-computable, $|b\rangle = |0\rangle$ and $M = |0\rangle\langle 0| \otimes I_{N/2}$: output a number that is within ϵ of $\langle x| M |x\rangle$ with probability $\geq 2/3$, where $|x\rangle$ is the unit vector proportional to $A^{-1}|b\rangle$.

The algorithm presented in our paper can be used to solve this problem with a small amount of overhead. By producing $|x\rangle$ up to trace distance $\epsilon/2$ in time $\tilde{O}(\log(N)\kappa^2 s^2/\epsilon)$, we can obtain a sample of a bit which equals one with probability μ with $|\mu - \langle x| M |x\rangle| \leq \epsilon/2$. Since the variance of this bit is $\leq 1/4$, taking $1/3\epsilon^2$ samples gives us a $\geq 2/3$ probability of obtaining an estimate within $\epsilon/2$ of μ. Thus quantum computers can solve the matrix inversion estimation problem in time $\tilde{O}(\log(N)\kappa^2 s^2/\epsilon^3)$.

We can now show that the error dependence of our algorithm cannot be substantially improved.

Theorem A.6. *1. If a quantum algorithm exists for the matrix inversion estimation problem running in time $\text{poly}(\kappa, \log(N), \log(1/\epsilon))$ then* **BQP**=**PP**.

2. No relativizing quantum algorithm for the matrix inversion estimation problem can run in time $N^\alpha \,\text{poly}(\kappa)/\epsilon^\beta$ unless $\alpha + \beta \geq 1$.

Proof. 1. A complete problem for the class **PP** is to count the number of satisfying assignments to a SAT formula. Given such formula ϕ, a quantum circuit can apply it on a superposition of all 2^n assignments for variables, generating the state

$$\sum_{z_1,\ldots,z_n \in \{0,1\}} |z_1, \ldots, z_n\rangle \, |\phi(z_1, \ldots z_n)\rangle \, .$$

The probability of obtaining 1 when measuring the last qubit is equal to the number of satisfying truth assignments divided by 2^n. A matrix inversion estimation procedure which runs in time $\operatorname{poly}\log(1/\epsilon)$ would enable us to estimate this probability to accuracy 2^{-2n} in time $\operatorname{poly}(\log(2^{2n})) = \operatorname{poly}(n)$. This would imply that $\mathbf{BQP} = \mathbf{PP}$ as required.

2. Now assume that $\phi(z)$ is provided by the output of an oracle. Let C denote the number of $z \in \{0,1\}^n$ such that $\phi(z) = 1$. From [22], we know that determining the parity of C requires $\Omega(2^n)$ queries to ϕ. However, exactly determining C reduces to the matrix inversion estimation problem with $N = 2^n$, $\kappa = O(n^2)$ and $\epsilon = 2^{-n-2}$. By assumption we can solve this in time $2^{(\alpha+\beta)n} \cdot \operatorname{poly}(n)$, implying that $\alpha + \beta \geq 1$.

□