Quantum Algorithm for Linear Systems of Equations

Aram W. Harrow,[1] Avinatan Hassidim,[2] and Seth Lloyd[3]

[1]Department of Mathematics, University of Bristol, Bristol, BS8 1TW, United Kingdom
[2]Research Laboratory for Electronics, MIT, Cambridge, Massachusetts 02139, USA
[3]Research Laboratory for Electronics and Department of Mechanical Engineering, MIT, Cambridge, Massachusetts 02139, USA

(Received 5 July 2009; published 7 October 2009)

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 does not 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, $N \times N$ and has condition number κ, the fastest known classical algorithms can find $\vec{x}$ and estimate $\vec{x}^\dagger M\vec{x}$ in time scaling roughly as $N\sqrt{\kappa}$. Here, we exhibit a quantum algorithm for estimating $\vec{x}^\dagger M\vec{x}$ whose runtime is a polynomial of $\log(N)$ and κ. Indeed, for small values of κ [i.e., poly $\log(N)$], we prove (using some common complexity-theoretic assumptions) that any classical algorithm for this problem generically requires exponentially more time than our quantum algorithm.

DOI: 10.1103/PhysRevLett.103.150502 PACS numbers: 03.67.Ac, 02.10.Ud, 89.70.Eg

Introduction.—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 so far found limited use outside the domain of quantum mechanics. This Letter presents a quantum algorithm to estimate features of the solution of a set of linear equations. Compared to classical algorithms for the same task, our algorithm can be as much as exponentially faster.

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. In other cases, such as when discretizing partial differential equations, the linear equations may be implicitly defined and thus far larger than the original description of the problem. For a classical computer, even to approximate the solution of Nlinear equations in N unknowns in general requires time that scales at least as N. 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 quantum computer can approximate the value of such a function in time which scales logarithmically in N, and polynomially in the condition number (defined below) and desired precision. The dependence on N is exponentially better than what is achievable classically, while the dependence on condition number is comparable, and the dependence on error is worse. Typically, the accuracy required is not very large. However, the condition number often scales with the size of the problem, which presents a more serious limitation of our algorithm. Coping with large condition numbers has been studied extensively in the context of classical algorithms. In the discussion section, we will describe the applicability of some of the classical tools (pseudoinverses, preconditioners) to our quantum algorithm.

We sketch here the basic idea of our algorithm and then discuss it in more detail in the next section. Given a Hermitian $N \times N$ matrix A, and a unit vector $\vec{b}$, suppose we would like to find $\vec{x}$ satisfying $A\vec{x} = \vec{b}$. (We discuss later questions of efficiency as well as how the assumptions we have made about A and $\vec{b}$ can be relaxed.) First, the algorithm represents $\vec{b}$ as a quantum state $|b\rangle = \sum_{i=1}^{N} b_i|i\rangle$. Next, we use techniques of Hamiltonian simulation [3,4] to apply e^{iAt} to $|b\rangle$ for a superposition of different times t. This ability to exponentiate A translates, via the well-known technique of phase estimation [5,6], 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 $\sum_{j=1}^{N} \beta_j|u_j\rangle|\lambda_j\rangle$, where u_j is the eigenvector basis of A, and $|b\rangle = \sum_{j=1}^{N} \beta_j|u_j\rangle$. We would then like to perform the linear map taking $|\lambda_j\rangle$ to $C\lambda_j^{-1}|\lambda_j\rangle$, where C is a normalizing constant. As this operation is not unitary, it has some probability of failing, which will enter into our discussion of the runtime below. After it succeeds, we uncompute the $|\lambda_j\rangle$ register and are left with a state proportional to $\sum_{j=1}^{N} \beta_j\lambda_j^{-1}|u_j\rangle = A^{-1}|b\rangle = |x\rangle$.

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. Our algorithms will generally assume that the sin-

gular values of A lie between $1/\kappa$ and 1; equivalently, $\kappa^{-2}I \leq A^\dagger A \leq I$. In this case, our algorithm uses roughly $O(\kappa^2 \log(N)/\epsilon)$ steps to output a state within distance ϵ of $|x\rangle$. Therefore, the greatest advantage our algorithm has over classical algorithms occurs when both κ and $1/\epsilon$ are poly $\log(N)$, in which case it achieves an exponential speedup.

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, often 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 [7]. Consider a stochastic process where the state of system at time t is described by the N-dimensional vector x_t and evolves according to the recurrence relation $\vec{x}_t = A\vec{x}_{t-1} + \vec{b}$. The stable state of this distribution is given by $|x\rangle = (I - A)^{-1}|b\rangle$. Let $\vec{x}'_t = A'\vec{x}'_{t-1} + \vec{b}'$ and $|x'\rangle = (I - A')^{-1}|b'\rangle$. To know if $|x\rangle$ and $|x'\rangle$ are similar, we perform the SWAP test between them [8]. We note that classically finding out if two probability distributions are similar requires at least $\Omega(\sqrt{N})$ samples [9].

The strength of the algorithm is that it works only with $O(\log N)$-qubit registers, and never has to write down all of A, $\vec{b}$, or $\vec{x}$. In situations (detailed below) where the Hamiltonian simulation and our nonunitary step incur only poly $\log(N)$ overhead, this means our algorithm takes exponentially less time than a classical computer would need even to write down the output. In that sense, our algorithm is related to classical Monte Carlo algorithms, which achieve dramatic speedups by working with samples from a probability distribution on N objects rather than by writing down all N components of the distribution. However, while these classical sampling algorithms are powerful, we will prove that in fact *any* classical algorithm requires in general exponentially more time than our quantum algorithms to perform the same matrix inversion task.

Outline.—The rest of the Letter proceeds by first describing our algorithm in detail, analyzing its runtime and comparing it with the best known classical algorithms. Next, we prove (modulo some complexity-theoretic assumptions) hardness results for matrix inversion that imply both that our algorithm's runtime is nearly optimal, and that it runs exponentially faster than any classical algorithm. We conclude with a discussion of applications, generalizations, and extensions.

Related work.—Previous papers gave quantum algorithms to perform linear algebraic operations in a limited setting [10]. Our work was extended by Ref. [11] to solving nonlinear differential equations.

Algorithm.—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. [3] shows how to simulate e^{iAt} in time

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

where the $\tilde{O}$ suppresses more slowly growing terms (described in Ref. [12]). If A is not Hermitian, define

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

As $\tilde{A}$ is Hermitian, we can solve the equation

$$\tilde{A}\,\vec{y} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}$$

to obtain

$$y = \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix}.$$

Applying this reduction if necessary, the rest of the Letter 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. [13] to prepare $|b\rangle$. Alternatively, our algorithm could be a subroutine in a larger quantum algorithm of which some other component is responsible for producing $|b\rangle$.

The next step is to decompose $|b\rangle$ in the eigenvector basis, using phase estimation [5,6]. Denote by $|u_j\rangle$, the eigenvectors of A (or equivalently, 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 Ref. [6]) to minimize a certain quadratic loss function which appears in our error analysis (see Ref. [12] for details).

Next, we apply the conditional Hamiltonian evolution $\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| \otimes e^{iA\tau t_0/T}$ on $|\Psi_0\rangle \otimes |b\rangle$, 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

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

Adding a qubit and rotating conditioned on $|\tilde{\lambda}_k\rangle$ yields

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

where C is chosen to be $O(1/\kappa)$. We now undo the phase estimation to uncompute the $|\tilde{\lambda}_k\rangle$. 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

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

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

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

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

Runtime and error analysis.—We present an informal description of the sources of error; the exact error analysis and runtime considerations are presented in Ref. [12]. Performing the phase estimation is done by simulating e^{iAt}. Assuming that A is s sparse, this can be done with error ϵ in time proportional to $ts^2(t/\epsilon)^{o(1)}=:\tilde{O}(ts^2)$.

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 [14], we find that $O(\kappa)$ repetitions are sufficient. Putting this all together, we obtain the stated runtime of $\tilde{O}(\log(N)s^2\kappa^2/\epsilon)$.

Classical matrix inversion algorithms.—To put our algorithm in context, one of the best general-purpose classical matrix inversion algorithms is the conjugate gradient method [15], 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.

The complexity of matrix inversion.—We will 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. As a result, if classical computers could estimate quantities of the form $\vec{x}^\dagger M\vec{x}$ in $\operatorname{poly}(\log N,\kappa,1/\epsilon)$ time, then any $\operatorname{poly}(n)$-gate quantum circuit could be simulated by a $\operatorname{poly}(n)$-time classical algorithm. Such a simulation is strongly conjectured to be false, and is known to be impossible in the presence of oracles [16].

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 runtime could be made polylogarithmic in κ, then any problem solvable on n qubits could be solved in $\operatorname{poly}(n)$ time (i.e., BQP = PSPACE), a highly unlikely possibility. 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 $\operatorname{poly}\log(1/\epsilon)$ would imply that BQP includes PP, and even minor improvements would contradict oracle lower bounds [17].

The reduction.—We now present the key reduction from simulating a quantum circuit to matrix inversion. Consider 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, corresponding to the observable $M=|0\rangle\langle 0|\otimes I^{\otimes n-1}$.

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

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

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

$$A^{-1}=\sum_{k\geq 0}U^k e^{-k/T}. \quad (5)$$

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. Using these techniques, it is possible to show that matrix inversion is BQP-complete. Full details of the proof are given in the supplementary online material [12].

Discussion.—There are a number of ways to extend our algorithm and relax the assumptions we made while presenting it. We will discuss first how to invert a broader class of matrices and then consider measuring other features of $\vec{x}$ and performing operations on A other than inversion.

Certain nonsparse A can be simulated and therefore inverted; see Ref. [4] for techniques and examples. It is also possible to invert nonsquare matrices, using the reduction presented from the non-Hermitian case to the Hermitian one.

The most important challenge in applying our algorithm is controlling the scaling of κ with N. In the worst case, κ can scale exponentionally with N, although this is not robust against small perturbations in A [18,19]. More often, κ is polynomial in N, in which case our algorithm may not outperform classical algorithms, or may offer only polynomial speedups. Finally, the ideal situation for our algorithm is when κ is polylogarithmic in N, as is the case with finite element models that use a fixed lattice spacing and a growing number of dimensions ([20], Section 9.6).

Given a matrix A with large condition number, our algorithm can also choose to invert only the part of $|b\rangle$ which is in the well-conditioned part of A (i.e., the subspace spanned by the eigenvectors with large eigenvalues). 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 the size of the ill-conditioned part, or to handle it in any other way she wants. If A is not invertible and $1/\kappa$ is taken to be smaller than the smallest nonzero eigenvalue of A, then this procedure can be used to compute the pseudoinverse of A.

Another method that is often used in classical algorithms to handle ill-conditioned matrices is to apply a preconditioner [21,22]. If we have a method of generating a preconditioner matrix B such that $\kappa(BA)$ is smaller than $\kappa(A)$, then we can solve $A\vec{x} = \vec{b}$ by instead solving the possibly easier matrix inversion problem $(BA)\vec{x} = B\vec{b}$. Further, if A and B are both sparse, then BA 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 appropriate nk-qubit observable on the state $|x\rangle^{\otimes k}$. Alternatively, one can use our algorithm to generate a quantum analogue of Monte Carlo calculations, 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 Refs. [4,11,23].

We thank the W. M. Keck Foundation for support, and A. W. H. thanks them as well as MIT for hospitality while this work was carried out. A. W. H. was also funded by the U.K. EPSRC grant "QIP IRC." S. L. thanks R. Zecchina for encouraging him to work on this problem. We are grateful as well to R. Cleve, D. Farmer, S. Gharabian, J. Kelner, S. Mitter, P. Parillo, D. Spielman, and M. Tegmark for helpful discussions.

[1] P. W. Shor, *Proc. of the 35th FOCS* (IEEE, New York, 1994), pp. 124–134.
[2] S. Lloyd, Science **273**, 1073 (1996).
[3] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders, Commun. Math. Phys. **270**, 359 (2007).
[4] A. M. Childs, arXiv:0810.0312.
[5] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, arXiv:quant-ph/9708016.
[6] A. Luis and J. Peřina, Phys. Rev. A **54**, 4564 (1996).
[7] D. G. Luenberger, *Introduction to Dynamic Systems: Theory, Models, and Applications* (Wiley, New York, 1979).
[8] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, Phys. Rev. Lett. **87**, 167902 (2001).
[9] P. Valiant, *Proc. of the 40th Annual STOC* (ACM, New York, 2008), pp. 383–392.
[10] A. Klappenecker and M. Rotteler, Phys. Rev. A **67**, 010302(R) (2003).
[11] S. K. Leyton and T. J. Osborne, arXiv:0812.4423.
[12] See EPAPS Document No. E-PRLTAO-103-055942 for detailed proofs of the claims in our manuscript. For more information on EPAPS, see http://www.aip.org/pubservs/epaps.html.
[13] L. Grover and T. Rudolph, arXiv:quant-ph/0208112.
[14] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, *Quantum Amplitude Amplification and Estimation*, Contemporary Mathematics Series Millenium Volume 305 (AMS, New York, 2002).
[15] J. R. Shewchuk, Technical Report No. CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, March 1994.
[16] D. R. Simon, SIAM J. Comput. **26**, 1474 (1997).
[17] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Phys. Rev. Lett. **81**, 5442 (1998).
[18] A. Sankar, D. A. Spielman, and S. H. Teng, SIAM J. Matrix Anal. Appl. **28**, 446 (2006).
[19] T. Tao and V. Vu, arXiv:0805.3167.
[20] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods* (Springer-Verlag, New York, 2008).
[21] K. Chen, *Matrix Preconditioning Techniques and Applications* (Cambridge Univ. Press, Cambridge, U.K., 2005).
[22] D. A. Spielman and S. H. Teng, arXiv:cs.NA/0607105.
[23] L. Sheridan, D. Maslov, and M. Mosca, J. Math Phys. A **42**, 185302 (2009).