Quantum algorithm for linear systems of equations

May 13, 2010

Abstract

This is a report accompanying the seminar given by me on 10 May 2010 at IISER Kolkata. Solving linear systems of equations is one of the most common problems that is used by itself and as a component in more complex problems. Given a matrix A and a vector $\vec{b}$, we have to find a vector $\vec{x}$ such that $A\vec{x} = \vec{b}$. Often we do not need the full vector $\vec{x}$, but instead only a function of it (such as average), denoted by 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 a sparse $N \times N$ matrix and has condition number κ (defined later), there is a quantum algorithm which can find $\vec{x}$ in $\text{poly}(\log N, \kappa)$ time which is exponentially better than the best classical algorithm.

1 Introduction

Solution of a set of linear equations is a well-known problem. One of the earliest descriptions of a solution to the problem has been found in the eighth chapter, *Rectangular Array* of the Chinese mathematical text *Jiuzhang suanshu* or *The Nine Chapters on the Mathematical Art*, dating from approximately 150 BCE. In 1809, Carl Friedrich Gauss independently found a method (subsequently known as Gauss' elimination method) which was published in the *Theory of Motion of Heavenly Bodies*. Gaussian elimination takes $O(N^3)$ time, where N is the size of A. An alternative is to use LU decomposition, which involves decomposing the matrix A into a lower triangular L and upper triangular U; then the system of equations can be easily solved.

Most quantum algorithms have been of three types - using the quantum fourier transform (Shor's algorithm), search algorithms (Grover) and simulation of quantum systems (evolution of Hamiltonians). The algorithm presented in the paper by Harrow, Hassidim and Lloyd gives an exponential speedup over the classical case when trying to find the solution vector $\vec{x}$, provided that we only want to perform an expectation value measurement on it. Classical algorithms are at least polynomial in N, whereas this algorithm is logarithmic in N.

The basic idea of the algorithm is given below and discussed in detail in the next section. Given a Hermitian $N \times N$ matrix A, and a unit vector $\vec{b}$, we would like to find $\vec{x}$ satisfying $A\vec{x} = \vec{b}$. First, the algorithm represents $\vec{b}$ as a quantum state $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$. Next, techniques of Hamiltonian simulation[3, 4] are used to apply e^{iAt} to $|b\rangle$ for a superposition of different times t. Using phase estimation and the exponentiation of A we can [5, 6, 7] decompose $|b\rangle$ in the eigenbasis of A and find the corresponding eigenvalues λ_j. The state of the system at 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 $|b\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle$. Then we perform a linear map taking $|\lambda_j\rangle$ to $C\lambda_j^{-1} |\lambda_j\rangle$, where C is a normalizing constant. This operation is not unitary and has a probability of failure which

will be discussed later. Then we can uncompute the $|\lambda_j\rangle$ register and get a state proportional to $\sum_{j=1}^{N} \beta_j \lambda_j^{-1} |u_j\rangle = A^{-1} |b\rangle = |x\rangle$.

As previously discussed, the algorithm depends upon the condition number of the matrix A denoted by κ. As the condition number becomes higher, A becomes closer to a matrix which cannot be inverted, and the solutions become less stable. Formally a condition number signifies the rate of change of the solution vector $\vec{x}$ as $\vec{b}$ is changed.

Let e be the error in b. Ratio of relative error in solution to relative error in b is

$$\frac{\|A^{-1}e\|/\|A^{-1}b\|}{\|e\|/\|b\|}$$

This is easily transformed to

$$(\|A^{-1}e\|/\|e\|) \cdot (\|b\|/\|A^{-1}b\|)$$

The maximum value (for nonzero b and e) is easily seen to be the product of the two operator norms:

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|$$

The operator norm is defined as $\sup \forall x \|Ax\|/\|x\|$.

A matrix with a high condition number is said to be "ill-conditioned." The algorithm assumes that the singular values of A lie between $1/\kappa$ and 1; equivalently $\kappa^{-2} I \leq A^\dagger A \leq I$. In that case, the runtime scales as $\kappa^2 \log(N)/\epsilon$, where ϵ is the additive error achieved in the output state $|x\rangle$. The greatest advantage of the algorithm compared to classical algorithms occurs when both κ and $1/\epsilon$ are poly $\log(N)$, in which case it achieves an exponential speedup. Some techniques will be discussed later for handling ill-conditioned matrices.

The algorithm gives us a representation of the solution $\vec{x}$ in the form of $|x\rangle$, from which we can get expectation values of an operator M, which may be varied to extract information about normalization, weights in different parts of the state space, moments, etc. One of the advantages 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 $\vec{x}$.

Later work which builds upon this algorithm is a paper on solving non-linear differential equations [12].

2 Algorithm

This section has 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 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 [13]). If A is not Hermitian, we can 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 algorithm assumes that A is Hermitian.

There also needs to be 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. [14] to prepare $|b\rangle$. Alternatively, the algorithm could be a part 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, 7]. If we 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 [5, 7]) to minimize a certain quadratic loss function which appears in the error analysis (see [13] 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 $t_0 = O(\kappa/\epsilon)$. Performing a Fourier transform on 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

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

Then we add an ancilla qubit and rotate based on $\left|\tilde{\lambda}_k\right\rangle$ to yield

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

where $C = O(1/\kappa)$. The phase estimation is then undone 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 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.

Run-time and error analysis The exact error analysis and runtime considerations are presented in [13]. 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 $ts^2(t/\epsilon)^{o(1)} =: \tilde{O}(ts^2)$.

Most of the error comes from phase estimation. This step errs by $O(1/t_0)$ in estimating λ, which results in 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, the probability of getting $|1\rangle$ is considered. Since $C = O(1/\kappa)$ and $\lambda \leq 1$, this probability is at least $\Omega(1/\kappa^2)$. Using amplitude amplification [15], it can be shown that $O(\kappa)$ repetitions are sufficient. Putting this all together, the runtime of the algorithm is $\tilde{O}\left(\log(N)s^2\kappa^2/\epsilon\right)$.

3 Optimality

Classical matrix inversion algorithms One of the best general-purpose classical matrix inversion algorithms is the conjugate gradient method [16], 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 $O(Ns\kappa\log(1/\epsilon))$.)

This algorithm can find out the expectation value of M; so 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 the quantum algorithm could be improved, say to achieve error ϵ in time proportional to $\operatorname{poly}\log(1/\epsilon)$. Both of these are not possible as is proved by using arguments from complexity theory. This is shown by proving that matrix inversion can be used to simulate a general quantum computation.

The complexity of matrix inversion It is shown 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. It is mostly likely that such a simulation is not possible and is known to be impossible in the presence of oracles [17].

The reduction from a general quantum circuit to a matrix inversion problem also implies that the 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 $\operatorname{poly}(n)$ time (i.e. **BQP**=**PSPACE**), which is highly unlikely.

The reduction In this part, the reduction from simulating a quantum circuit to matrix inversion is shown. 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 we adjoin an ancilla register of dimension $3T$ and define a unitary

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

U has been chosen suchx 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 the second register is left 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 the above claims about the difficulty of improving the algorithm.

4 Conclusion

The algorithm can be extended in a number of ways. The paper refers to partial inversion of ill-conditioned matrices. Another method is to apply a preconditioner [19]. The preconditioner is (in this case) a sparse matrix B which when multiplied with A reduces the overall κ, thus making it easier to calculate the solution.

An extreme generalisation of the algorithm is not to invert the matrices at all. Instead, it can compute $f(A) |b\rangle$ for any computable f. Depending on the degree of nonlinearity of f, non-trivial tradeoffs between accuracy and efficiency arise. Some similar ideas are considered in [20, 4, 12].

Acknowledgements. I thank Dr. Chiranjib Mitra for suggesting this topic.

References

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

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

[3] 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.

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

[5] A. Luis and J. Peřina. Optimum phase-shift estimation and the quantum description of the phase difference. *Phys. Rev. A*, 54(5):4564–4570, Nov 1996.

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

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

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

[9] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf. Quantum fingerprinting. *Phys. Rev. Lett.*, 87(16):167902–167902, 2001.

[10] P. Valiant. Testing symmetric properties of distributions. In *Proceedings of the 40th Annual ACM Symposium on Theory of computing (STOC)*, pages 383–392. ACM Press New York, NY, USA, 2008.

[11] A. Klappenecker and M. Rotteler. Quantum Physics Title: Engineering Functional Quantum Algorithms. *Phys. Rev. A*, 67:010302(R), 2003.

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

[13] A.W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for solving linear systems of equations, 2008. arXiv:0811.3171.

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

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

[16] 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.

[17] Daniel R. Simon. On the power of quantum computation. *SIAM J. Comp.*, 26:1474–1483, 1997.

[18] 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.

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

[20] L. Sheridan, D. Maslov, and M. Mosca. Approximating fractional time quantum evolution, 2009. arXiv:0810.3843.

[21] D. Aharonov and A. Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the 35th Annual ACM Symposium on Theory of computing (STOC)*, pages 20–29. ACM Press New York, NY, USA, 2003. arXiv:quant-ph/0301023.

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

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

[24] 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.