Quantum algorithm for linear systems of equations

Final Report

Mayank Sharma

Solving linear systems of equations is ubiquitous in all areas of science and engineering. With rapidly growing data sets, such a task can be intractable for classical computers, as the best known classical algorithms require a time proportional to the number of variables N, or what we know a poly(N)-time. The quantum algorithm for linear systems of equations shows that quantum computers could solve linear systems in a time scale of order Log(N) giving an exponential speedup over classical computers.

Introduction

Historically,there are three most prominent quantum algorithms that can achieve an exponential speedup over classical computers. The first one is quantum simulation of complex systems proposed by Richard P. Feynman, "Simulating physics with computers" which was published in the 1980s. The second one is by P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring" also know as Shors Algorithm for factoring large numbers which was published in 1994. The third was by Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd, "Quantum algorithm for linear systems of equations". Published in 2009 , it would be fair to say that this might be the most practically useful quantum algorithm so far as the problem of solving linear equations is ubiquitous in virtually all areas of science and engineering .
We come across such linear equations daily in the field of computing and an improvement such as the Quantum Algorithm for linear systems of equations will increase the processing speed tremendously if put into use. It would raise our ability to solve problems that are nearly impossible right now because our current processing power simply makes it unfeasible to solve them. It would allow us to change the way we solve real time problems and open up new avenues for research that we are currently ill equipped to handle.

The quantum algorithm for linear systems of equations, by Aram Harrow, Avinatan Hassidim, and Seth Lloyd estimates the result of a scalar measurement on the solution vector to a given linear space of equations. By getting an approximation instead of an exact solution this algorithm is able to speed up further processing in some cases.

Provided the linear system is a sparse matrix and has a low condition number k , and that the user is interested in the result of a scalar measurement on the solution vector, instead of the values of the solution vector itself, then the algorithm has a runtime of

$O(log(N)k^2)$

, where N is the number of variables in the linear system. This provides us with an exponential improvement over the fastest classical algorithm , with runs in

$O(Nk)$

Given a Hermitian NxN matrix A, and a unit vector $\vec{b}$, suppose we would like to find $\vec{x}$ satisfying $A\vec{x} = \vec{b}$. This algorithm assumes that the user is not interested in the values of $\vec{x}$ itself, but rather the result of applying some operator M onto x, $< x|M|x >$. With this assumption in mind we use the Quantum Algorithm to find $< x|M|x >$ faster than any classical approach.

Preliminary Considerations

An important factor in the performance of the matrix inversion algorithm is k, 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 $1/k$ and 1. Equivalently $k^{-2}I <= A^{\dagger}A <= I$.

In this case, our runtime will scale as $k^2 log(N)/\epsilon$, where ϵ is the additive error achieved in the output state $|x >$. Therefore, the greatest advantage our algorithm has over classical algorithms occurs when both k and $1/\epsilon$ are $polyLog(N)$, in which case it achieves an exponential speedup.

The algorithm requires that the matrix A be Hermitian so that it can be converted into a unitary operator. In the case where A is not Hermitian, we must perform the following reduction.

C $= \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$

The condition number is a crucial parameter in the algorithm. One possible method of handling ill-conditioned matrices is as follows.

We will define the well-conditioned part of A to be the span of the eigenspaces corresponding to eigenvalues $>= 1/\kappa$ 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 $<= 1/\kappa'$ as ill-conditioned, and interpolates between these two behaviors when $1/\kappa' < |\lambda| < 1/\kappa$

Procedure Overview

Step 1: The algorithm represents $\vec{b}$ as a quantum state

$|b >= \Sigma_{i=0}^{N} b_i |i >$.

Step 2: Hamiltonian simulation techniques are used to apply the unitary operator e^{iAt} to $|b >$ for a superposition of different times t. The ability to decompose $|b >$ into the eigenbasis of A and to find the corresponding eigenvalues λ_j is done by the use of quantum phase estimation . The state of the system after this stage is approximated to

$\Sigma_{j=1}^{N}\beta_j|u_j>|\lambda_j>$

where u_j is the eigenvector basis of A, and

$|b>=\Sigma_{j=1}^{N}\beta_j|u_j>$

Step 3: Perform the linear map taking $|\lambda_j>$ to $C\lambda_j^{-1}|\lambda>$, where C is a normalizing constant. In probability theory, a normalizing constant is a constant by which an everywhere non-negative function must be multiplied so the area under its graph is 1, e.g., to make it a probability density function or a probability mass function.
As this operation is not unitary, it has some probability of failing and so we repeat it a number of times till it succeeds. We uncompute the $|\lambda_j>$ register and are left with a state proportional to:

$\Sigma_{i=0}^{N}\beta_i\lambda_j^{-1}|u_j>=A^{-1}|b>=|x>$

Result: This procedure yields a quantum-mechanical representation $|x>$ of the desired vector $\vec{x}$. To read out all the components of $\vec{x}$ would require one to perform the procedure at least N times,however, often we are interested not in $\vec{x}$ itself, but in some expectation value $\vec{x^T}M\vec{x}$, where M is some linear operator. By mapping M to a quantum-mechanical operator, and performing the quantum measurement corresponding to M, we obtain an estimate of the expectation value $<x|M|x>=\vec{x^T}M\vec{x}$, 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.

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 $\vec{x}$ Thus , giving it an edge over classical algorithms.

Algorithm

Step 1: We want to transform a given Hermitian matrix A into a unitary operator of the form e^{iAt} which we can then apply wherever we need. This is

possible if A is s-sparse, which means it has a maximum of s non zero entries in per row, and is row computable , which means given a row index these entries can be computed in time O(s).
Assuming these two conditions the Efficient Quantum Algorithms for Simulating Sparse Hamiltonians given by D.W. Berry, G. Ahokas, R. Cleve, and B.C. Sanders show us how to simulate e^{iAt} in time

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

where the $\check{O}$ negates slow growing terms. If A is not Hermitian , we take

C = $\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$

As C is Hermitian , we can solve the equation $C\vec{y} = \begin{bmatrix} \vec{b} \\ 0 \end{bmatrix}$ to get y= $\begin{bmatrix} 0 \\ \vec{x} \end{bmatrix}$

Step 2:In a lot of cases our algorithm will be a subroutine in a bigger quantum algorithm and some other part may be responsible for the preparation of $|b>$. In such cases we can simply utilize the incoming variable. If this is not the case, we use the method given for Creating superpositions that correspond to efficiently integrable probability distributions by L. Grover and T. Rudolph to prepare $|b>$.

Step 3: Use Phase estimation as given in Optimum phase-shift estimation and the quantum description of the phase difference by A. Luis and J. Perina to decompose $|b>$ in the eigenvector basis. We will write the eigenvectors of A as $|u_j>$ and the corresponding eigenvalues by λ_j. Assume now

$$|\psi_0>:= \sqrt{\frac{2}{T}}\Sigma_{T=0}^{T-1} sin\frac{\pi\tau+\frac{1}{2}}{T}|\tau>$$

Step 4: We apply the conditional Hamiltonian evolution

$$\Sigma_{T=0}^{T-1}|\tau><\tau|^C \otimes e^{iA\tau t_0/T}$$

on $|\psi_0>^c \otimes|b>$, where $t_0 = O(\kappa/\epsilon)$.

Step 5: Fourier transforming the first register gives the state

$\Sigma_{j=1}^{N}\Sigma_{k=0}^{T-1}\alpha_{\kappa}|j\beta j|\kappa > |u_j >$

where $|\kappa >$ are the Fourier basis states, and $\alpha_{\kappa|j}|$ is large if and only if λ_j approaches $\frac{2\pi\kappa}{t_0}$. Defining $\check{\lambda_k} = \frac{2\pi\kappa}{t_0}$ we can obtain the following

$\Sigma_{j=1}^{N}\Sigma_{k=0}^{T-1}\alpha_{\kappa}|j\beta j|\check{\lambda}_k > |u_j >$

Step 6: Adding an ancilla qubit and rotating conditioned on $\check{\lambda_k}$ we get

$\Sigma_{j=1}^{N}\Sigma_{k=0}^{T-1}\alpha_{\kappa}|j\beta j|\check{\lambda}_k > |u_j > (\sqrt{1-\frac{C^2}{\check{\lambda}_k^2}}|0> + \frac{C}{\check{\lambda}_k}|1>)$

where C$= O(1/\kappa)$.

Step 7: We now undo the phase estimation to uncompute the $\check{\lambda}_k$. If the phase estimation were perfect, we would have $\alpha_{\kappa|j}$ =1 if $\check{\lambda}_k$ =λ_j , and 0 otherwise. Assuming the perfect state of phase estimation , we get

$\Sigma_{j=1}^{N}|\beta j|u_j > (\sqrt{1-\frac{C^2}{\check{\lambda}_k^2}}|0> + \frac{C}{\check{\lambda}_k}|1>)$

Step 8: To finish the inversion we measure the last qubit. Conditioned on seeing 1, we have the state

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

which corresponds to $\Sigma_{j=1}^{N}\beta_j\lambda_j^{-1}|u_j >$ up to normalization. We can determine the normalization factor from the probability of obtaining 1.

Step 9: Finally, we make a measurement M whose expectation value $< x|M|X >$ corresponds to the feature of $\vec{x}$ that we wish to evaluate.

Run time analysis

1. Classical efficiency

   The best classical algorithm which produces the actual solution vector $\vec{x}$ is Gaussian elimination(also known as row reduction) which is an algorithm for solving systems of linear equations. It is usually understood as a sequence of operations performed on the corresponding matrix of coefficients. This method can also be used to find the rank of a matrix, to calculate the determinant of a matrix, and to calculate the inverse of an invertible square matrix. The method is named after Carl Friedrich Gauss $(1777-1855)$. This method provides the result in $O(N^3)$ time. If A is sparse and positive semi-definite, then the Conjugate Gradient method is used to find the solution vector $\vec{x}$. The conjugate gradient method is often implemented as an iterative algorithm, applicable to sparse systems that are too large to be handled by a direct implementation or other direct methods such as the Cholesky decomposition. Large sparse systems often arise when numerically solving partial differential equations or optimization problems. This can be found in $O(Ns\kappa)$ time by minimizing the quadratic function $|A\vec{x}-\vec{b}|^2$. When only an approximation of the of the solution vector $\vec{x}$ is needed, as is the case for the quantum algorithm for linear systems of equations, a classical computer can find an estimate of $\vec{x}^\dagger M\vec{x}$ in $O(N\sqrt{\kappa})$.

2. Quantum efficiency

   The quantum algorithm for solving linear systems of equations originally proposed by Harrow et al. was shown to run in $O(\kappa^2 \log N)$. The runtime of this algorithm was subsequently improved by Andris Ambainis to run in $O(\kappa \log^3 \kappa \log N)$ in his paper Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations by Adris Ambainis. Since the algorithm by Harrow, Hassidim and Lloyd maintains its logarithmic scaling only for sparse or low rank matrices, A quantum linear system algorithm for dense matrices by Leonard Wossnig, Zhikuan Zhao, Anupam Prakash extended the algorithm based on a quantum singular value estimation technique and provide a linear system algorithm for dense matrices which runs in $O(\sqrt{N}\log N\kappa^2)$time compared to the $O(N\log N\kappa^2)$ of the standard Harrow, Hassidim and Lloyd algorithm.

3. Optimum Conditions

   An important factor to be considered in the performance of the matrix inversion algorithm is the condition number κ of A , which represents the ratio of A's largest and smallest eigenvalues. As the κ increases, the ease with which the solution vector can be found using gradient descent methods such as the conjugate gradient method which is used in the Quantum algorithm for linear systems of equations decreases, as A becomes closer to a matrix which cannot be inverted and the solution vector becomes less stable. This algorithm assumes that all elements of the matrix A lie between $\frac{1}{\kappa}$ and 1, in which case the claimed run-time proportional to κ^2 will be achieved. Therefore, the advantage in run-time over classical algorithms is increased further when κ is a *poly*$(\log(N))$. If the run-time of the algorithm were made polylogarithmic in κ then problems solvable on n qubits could be solved in poly(n) time, causing the complexity class of bounded-error quantum polynomial time to be equal to PSPACE or polynomial amount of space.

Error analysis

When we take $t_0 = O(\kappa/\epsilon)$, we introduce an error of $<= \epsilon$ in the final state. The error in both the well-conditioned and the ill-conditioned cases is upper-bounded by $O(\kappa/t_0)$.

The Hamiltonian simulation, which becomes the greatest source of error, is done by simulating e^{iAt}. Assuming that A is sparse, this can be done with an error bounded by a constant ε , which will introduce an additive error in the output state $|x>$.

The phase estimation step gives an error of the order $O(\frac{1}{t_0})$ in estimating λ, which then gives a relative error of $O(\frac{1}{\lambda t_0})$ in λ^{-1}.
If $\lambda >= \frac{1}{k}$,taking $t_0 = O(\kappa\epsilon)$ introduces an error of ϵ in the final state. Thus it is required that the overall run time efficiency should be increased proportionally to $O(\frac{1}{\epsilon})$ to minimize error in the output state $|x>$.

Further Developments and Open Problems

In 2013 ,Experimental realization of quantum algorithm for solving linear systems of equations by Jian Pan, Yudong Cao, Xiwei Yao, Zhaokai Li, Chenyong Ju, Xinhua Peng, Sabre Kais, Jiangfeng Du was one of three experiments in parallel to to implement of a "proof of concept" by demonstrating real world results in using this algorithm with over 96% fidelity. These experiments were a first implementation of the algorithm.

The most recent work in the field would have to be A quantum linear system algorithm for dense matrices by Leonard Wossnig, Zhikuan Zhao and Anupam Prakash, which was published on the 20th of April, 2017 which gives us a quadratic speedup over the original Quantum algorithm for solving linear system of equations.
We are still not at the stage of implementing quantum computers in our daily lives, however, such works with their improvements to our understanding of quantum computing will most certainly help us in many facets of our lives.

Perhaps the most far-reaching generalization of the matrix inversion algorithm would be not to invert matrices at all. Instead, it could try and compute $f(A)|b>$ for any computable f. Depending on the degree of nonlinearity of f, nontrivial tradeoffs between accuracy and efficiency would of course arise and would need to be addressed.

Bibliography

1. P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring.
2. D.W.Berry, G. Ahokas, R. Cleve, and B.C. Sanders. Efficient Quantum Algorithms for Simulating Sparse Hamiltonians.
3. A. Luis and J. Perina. Optimum phase-shift estimation and the quantum description of the phase difference.
4. S. K. Leyton and T. J. Osborne. A quantum algorithm to solve nonlinear differential equations
5. A.W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for solving linear systems of equations
6. Experimental quantum computing to solve systems of linear equations by Cai et al
7. Quantum linear system algorithm for dense matrices by Leonard Wossnig, Zhikuan Zhao, Anupam Prakash