CR04 Report: Solving linear equations on a quantum computer

VU Thi Xuan

December 01, 2016

Abstract

In this report, we study a quantum algorithm for linear systems of equations due to the original work by Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd [9].

1 Introduction

Polynomial system solving which is given as problem 1 is an important problem with a huge application. It is crucial to have a efficient algorithm to solve this problem, especially when n is big. Classical algorithms for systems of linear equations run in time $\mathcal{O}(n^{\omega})$, where ω is the exponent of matrix multiplication. If instead of interesting in exact solutions for the systems of linear equations, one is interested in approximate solutions, one of the best classical algorithms is the conjugate gradient method [11]. In [11], when A is positive definite, the total run-time is $\mathcal{O}(ns\sqrt{\kappa}\log(1/\epsilon))$; if A is not positive definite, the total time is $\mathcal{O}(ns\kappa\log(1/\epsilon))$. Hereafter, s and κ are the sparselity and the condition number of the matrix A, respectively (see section 2 for more details); and ϵ is the final error of the algorithm.

Problem 1 (Linear systems solving)**.**
Input: *a $n \times n$ matrix A, an vector $\vec{b}$*
Output: *vector $\vec{x}$ satisfying $A\vec{x} = \vec{b}$.*

Quantum computers exploit quantum mechanical to perform computations in ways that classical computers cannot. For certain problems such as Shor's factoring large numbers algorithm, quantum algorithm provide exponential speedups over their classical counterpart. A natural question is can we construct an efficient quantum algorithm for problem 1 which is exponentially faster than what is possible with classical computation? Harrow, Hassidim and Lloyd [9] (HHL) proposed a quantum algorithm for obtaining certain information about solution $\vec{x}$ for a linear system $A\vec{x} = \vec{b}$. The input of HHL algorithm is a Hermitian matrix $A \in \mathbb{C}^{n\times n}$, and a unit vector $\vec{b}$. The authors also give the ways how to relax these conditions for A and $\vec{b}$ (see section 3.5). HHL algorithm is a surprising quantum algorithm with the run-time is $\tilde{\mathcal{O}}(\log(n)s^2\kappa^2/\epsilon)$. The greatest advantage of HHL algorithm comparing over the classical algorithms is when both κ and $1/\epsilon$ are poly $\log(n)$. However, HHL algorithm has some limits. The first one is we have not an efficient algorithm to present $\vec{b}$ as a quantum state $|b\rangle = \sum_{i=1}^{n} b_i |i\rangle$ in general (see the beginning of section 3). Therefore, HHL algorithm is used as a subroutine in a larger quantum algorithm of which some other component prepare $|b\rangle$. The other limit is one can read the values $x_1, ..., x_n$ from the quantum state $|x\rangle = \sum_{i=1}^{n} x_i |i\rangle$ in at least n times. For this reason, the application of the algorithm is limited to cases where one is interested in expectation value $\vec{x}^\dagger M \vec{x}$, where M is some linear operator.

Extended modifications quantum algorithm for problem 1 have been applied to other important problems such as solving linear differential equations [4], and machine learning [10].

The report is organized as follows. In the section 2, we recall some notations as well as some properties to understand the paper. We present HHL algorithm in the section 3. In this section, we make more precise details than the authors's description in [9]. Next, we give the algorithm's complexity, the optimality of algorithm and some discussions which proposed by authors. The report ends with conclusion in section 4. In this section, first we will summary the works provided by authors in their paper [9]; second we give some subsequent works to improve HHL algorithm without going in to details these works.

2 Preliminaries

In this section, we recall some definitions and some properties which are needed to understand the paper. In their algorithm, the authors consider that the input matrix A is s-sparse.

Definition 1. *A matrix is s-*sparse *if each row has at most s non-zero entries, and given a row index, these entries can be computed in $\mathcal{O}(s)$.*

Proposition 1. *Let A be an $n \times n$ matrix. Let $\{\lambda_j\}_{j=1}^n$ be the eigenvalues of A; and $\{u_j\}_{j=1}^n$ be the eigenvectors of A. Then,*

- *the inverse matrix, A^{-1}, has eigenvalues $\{\lambda_j^{-1}\}_{j=1}^n$ corresponding to the same eigenvectors, $\{u_j\}_{j=1}^n$,*
- *the matrix e^{iAt} has eigenvalues $\{e^{i\lambda_j t}\}_{j=1}^n$ corresponding to the same eigenvectors, $\{u_j\}_{j=1}^n$, and*
- *in addition, if $\{u_j\}_{j=1}^n$ are the eigenvectors normalized of A, then $A = \sum_{j=1}^{n} \lambda_j |u_j\rangle \langle u_j|$.*

The condition number, κ, is a crucial parameter in their algorithm. The *condition number* κ of the matrix A is the ratio between A's largest and smallest eigenvalues. That is $\kappa = \max_j |\lambda_j| / \min_j |\lambda_j|$. A larger condition number means that A becomes closer to a matrix which can not be inverted. Their algorithm will generally assume that $\kappa^{-1} \leqslant \lambda_j \leqslant 1$ for all j. The matrices in which $\kappa^{-1} \leqslant \lambda_j \leqslant 1$ for all j is *well-conditioned*.

The authors prove their algorithm is optimal by two different points of view. The first one is based on the complexity theory and the other one is based on oracles. To understand that, we recall some results in quantum complexity class.

Definition 2 (BQP). *A language L is in* BQP *if and only if there exists a polynomial-time uniform family of quantum circuits $\{Q_n\}_{n\in\mathbb{N}}$ such that*

- *For all $n \in \mathbb{N}$, Q_n takes n qubits as input and outputs 1.*
- *For all $x \in L$, $\mathbb{P}(Q_{|x|}(x) = 1) \geqslant \frac{2}{3}$.*
- *For all $x \notin L$, $\mathbb{P}(Q_{|x|}(x) = 0) \geqslant \frac{2}{3}$.*

Theorem 1. $\mathsf{P} \subseteq \mathsf{BPP} \subseteq \mathsf{BQP} \subseteq \mathsf{PP} \subseteq \mathsf{PSPACE}$.

We recommend [12] and [13] for the reader who wants to know more about quantum computational complexity.

3 Algorithm

We will, first, present the authors's algorithm in section 3.1; in section 3.2, we will go into the details of the algorithm. In section 3.3, we will give the complexity of the algorithm and the error analysis. The optimality of this algorithm and some discussions will be given in section 3.4 and section 3.5, respectively.

Given a Hermitian $n \times n$ matrix A, and a unit vector $\vec{b}$, we would like to find $\vec{x}$ such that $A\vec{x} = \vec{b}$. We will later show that, in section 3.5, how the assumptions the authors made about A and $\vec{b}$ can be relaxed. First, we need an efficient procedure to present $\vec{b}$ as a quantum state $|b\rangle = \sum_{i=1}^{n} b_i |i\rangle$. However, for this step, the authors assume that there exists oracle to efficiently prepare state $|b\rangle$. Equivalently, their algorithm is a subroutine in a larger quantum algorithm of which $|b\rangle$ is already produced by some other components. As before, we have seen that we can always write $A^{-1} = \sum_{j=1}^{n} \lambda_j^{-1} |u_j\rangle \langle u_j|$, where $\{\lambda_j\}_{j=1}^{n}$ are the eigenvalues and $\{u_j\}_{j=1}^{n}$ are the eigenvectors normalized of A, respectively. The main idea is if we can decompose $|b\rangle$ in the eigenbasis of A, namely $|b\rangle = \sum_{j=i}^{n} \beta_j |u_j\rangle$, and we can find the corresponding eigenvalues λ_j, the solution can be formally expressed as $|x\rangle = A^{-1} |b\rangle = (\sum_{j=1}^{n} \lambda_j^{-1} |u_j\rangle \langle u_j|)(\sum_{j=i}^{n} \beta_j |u_j\rangle) = \sum_{j=1}^{n} \beta_j \lambda_j^{-1} |u_j\rangle$. Here, we used the fact that $\langle u_j| |u_i\rangle = 1$ if $i = j$, and 0 otherwise. To obtain the decomposition of $|b\rangle$ and λ_j, the authors use techniques of Hamiltonian simulation to apply e^{iAt} to $|b\rangle$ for a superposition of different times t. The Hamiltonian evaluation is part of the well-known technique, *phase estimate algorithm*. Next the authors perform the linear map taking $|\lambda_j\rangle$ to $C\lambda_j^{-1} |\lambda_j\rangle$, where C is normalizing constant; and they uncompute the $|\lambda_j\rangle$. After all, we obtain $|x\rangle = \sum_{j=1}^{n} \beta_j \lambda_j^{-1} |u_j\rangle$ up to normalization.

This procedure yields a quantum mechanical representation $|x\rangle = \sum_{i=1}^{n} x_i |i\rangle$ of desired vector $\vec{x}$. The algorithm finds the solution $|x\rangle$ in time $\mathcal{O}(\log(n))$, yet, to read out all x_i still require $\mathcal{O}(n)$ times. The application of the algorithm is limited to cases where one is only interested in some expectation value $\vec{x}^{\dagger} M \vec{x}$, where M is some linear operator. A detailed complexity analysis for this algorithm which we will present in section 3.3 shows that the algorithm runs in $\mathcal{O}(\log(n)\kappa^2/\epsilon)$, where ϵ is the total error, which is given as a part of the input, to produce the output state $|x\rangle$.

3.1 Algorithm

Based on the discussions we gave at the beginning of this section, we present here the algorithm. We will give a more detailed description of the algorithm in the next subsection. During the algorithm, the authors introduce parameters $t_0 = \mathcal{O}(\kappa/\epsilon)$ and $C = \mathcal{O}(1/\kappa)$. In section 3.3, we will explain why t_0 and C can be taken to be $\mathcal{O}(\kappa/\epsilon)$ and $\mathcal{O}(1/\kappa)$, respectively. Let

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

for some large T which to be chosen latter in section 3.2 as T_H. The algorithm presented by the authors can be summarized as:

1. Represent $\vec{b}$ as $|b\rangle = \sum_{i=1}^{n} b_i |i\rangle$.

2. Prepare $|\Psi_0\rangle$ from $|0\rangle$ up to error ϵ_Ψ.

3. Apply the conditional Hamiltonian evolution $\sum_{\tau=0}^{T-1} |\tau\rangle \langle\tau| \otimes e^{iAt_0/T}$ on $|\Psi_0\rangle \otimes |b\rangle$ up to error ϵ_H.

4. Applying $QFT^\dagger$ on the first register gives the state

$$\sum_{j=1}^{n}\sum_{k=0}^{T-1} \alpha_{k|j}\beta_j \left|k\right\rangle \left|u_j\right\rangle ,$$

where $\alpha_{k|j} = \frac{\sqrt{2}}{T}\sum_{\tau=0}^{T-1} e^{i\frac{\tau}{T}(\lambda_j t_0 - 2\pi k)} \sin\frac{\pi(\tau+\frac{1}{2})}{T}$. Defining $\tilde{\lambda}_k := 2\pi k/t_0$, we can relabel $\left|k\right\rangle$ register to obtain

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

5. Adding an ancilla qubit and performing a rotation onto the ancilla qubit yields

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

6. Applying the inverse of the phase estimate to uncompute the $\left|\tilde{\lambda}_k\right\rangle$. If the phase estimation makes no error, we would have $\alpha_{k|j} = 1$ if $\tilde{\lambda}_k = \lambda_j$, and 0 otherwise. Assuming this, we obtain

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

7. Applying a measurement on the last qubit, when measured to be 1, we have the state

$$\sqrt{\frac{1}{\sum\limits_{j=1}^{n} C^2|\beta_j|^2|\lambda_j|^2}}\sum_{j=1}^{n}\beta_j\frac{C}{\lambda_j}\left|u_j\right\rangle .$$

which is the solution, $\left|x\right\rangle = \sum_{j=1}^{n}\beta_j\lambda_j^{-1}\left|u_j\right\rangle$, of the system up to normalization.

If one is only interested in some expectation value $\vec{x}^T M\vec{x}$, where M is some linear operator, taking a measurement M allows to compute $\left\langle x\right| M \left|x\right\rangle$.

3.2 Detailed description of the algorithm

In this subsection, we will present a detailed description for each step in the algorithm which is given in the previous subsection.

Step 1: If b_i and $\sum_{i=i_1}^{i_2}|b_i|^2$ are efficiently computable then we can use the procedure of [8] to prepare $\left|b\right\rangle = \sum_{i=1}^{n} b_i\left|i\right\rangle$. Otherwise, the authors assume that their algorithm is a subroutine in a larger quantum algorithm of which $\left|b\right\rangle$ is already produced by some other components.

Step 2: To prepare $\left|\Psi_0\right\rangle$ up to error ϵ_Ψ, the authors use [8].

Step 3: First, to simulate e^{iAt} for some $t \geqslant 0$, the authors use the algorithm of [3]. If A is s-sparse, $t \leqslant t_0$ and the simulation error is $\leqslant \epsilon_H$, then the time is

$$T_H = \mathcal{O}(\log(n)(\log^*(n))^2 s^2 t_0 9^{\sqrt{\log(s^2 t_0/\epsilon_H)}}) = \tilde{\mathcal{O}}(\log(n)s^2 t_0),$$

where the notation $\tilde{\mathcal{O}}(\cdot)$ indicates that $\log^*(n)$ and $9^{\sqrt{\log(s^2 t_0/\epsilon_H)}}$ factors are omitted.

Second, by applying the conditional Hamiltonian evolution $\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| \otimes e^{iAt_0/T}$ on $|\Psi_0\rangle \otimes |b\rangle$ up to error ϵ_H, we obtain

$$\begin{aligned}
(\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| \otimes e^{iAt_0/T})(|\Psi_0\rangle \otimes |b\rangle) &= \sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| |\Psi_0\rangle \otimes e^{iAt_0/T} |b\rangle , \\
&= \sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| \sqrt{\frac{2}{T}} \sum_{\tau=1}^{T-1} \sin \frac{\pi(\tau+\frac{1}{2})}{T} |\tau\rangle \otimes e^{iAt_0/T} |b\rangle , \\
&= \sum_{\tau=0}^{T-1} \sqrt{\frac{2}{T}} \sin \frac{\pi(\tau+\frac{1}{2})}{T} |\tau\rangle \otimes \sum_{j=1}^{n} e^{\frac{i\lambda_j t_0 \tau}{T}} |u_j\rangle\langle u_j| |b\rangle , \\
&= \sum_{\tau=0}^{T-1} \sqrt{\frac{2}{T}} \sin \frac{\pi(\tau+\frac{1}{2})}{T} |\tau\rangle \otimes \sum_{j=1}^{n} e^{\frac{i\lambda_j t_0 \tau}{T}} \beta_j |u_j\rangle .
\end{aligned}$$

Step 4: Applying the $QFT^\dagger$ on the $|\tau\rangle$ register yields

$$\sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin \frac{\pi(\tau+\frac{1}{2})}{T} \sum_{k=0}^{T-1} \frac{1}{\sqrt{T}} e^{-\frac{2\pi i \tau k}{T}} |k\rangle \otimes \sum_{j=1}^{n} e^{\frac{i\lambda_j t_0 \tau}{T}} \beta_j |u_j\rangle = \sum_{j=1}^{n} \sum_{k=0}^{T-1} \alpha_{k|j} \beta_j |k\rangle |u_j\rangle ,$$

where $\alpha_{k|j} = \sum_{\tau=0}^{T-1} \frac{\sqrt{2}}{T} e^{\frac{i\tau}{T}(\lambda_h t_0 - 2\pi k)} \sin \frac{\pi(\tau+\frac{1}{2})}{T}$. The authors give a bound $|\alpha_{k|j}|^2 \leqslant 64\pi^2/\delta^2$ whenever $|k - \lambda_j t_0/2\pi| \geqslant 1$, where $\delta := \lambda_j t_0 - 2\pi k$. Defining $\tilde{\lambda}_k := 2\pi k/t_0$, we can relabel $|k\rangle$ register to obtain

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

Step 5: We use the state $|\tilde{\lambda}_k\rangle$ to perform a rotation $R_y(\theta_j)$, where $\theta_j = 2\arcsin\left(\frac{C}{\tilde{\lambda}_k}\right)$, and $R_y(\theta) := \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}$ with $C = \mathcal{O}(1/\kappa)$. After that, we add an ancilla qubit on which the rotation $R_y(\theta_j)$ is performed, then the state becomes

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

Step 6 and *Step 7:* They are described as in the previous subsection.

3.3 Run-time and Error analysis

In this section, we give an analysis of the run-time and error probability of the algorithm.

First, let A be s-sparse, simulating e^{iAt} [3] up to error is $\leqslant \epsilon_H$ for some $t \geqslant 0$ can be done in time

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

Note that this is the only step where we require that A is sparse. Furthermore, using [8], we can prepare $|\Psi_0\rangle$ up to error ϵ_Ψ in time poly $\log(T_H/\epsilon_\Psi)$.

Next, we give the reasons why C and t_0 can be taken to be $\mathcal{O}(\kappa)$ and $\mathcal{O}(\kappa/\epsilon)$ respectively, where ϵ is an error in the final state. The parameter C is a normalization constant chosen to ensure rotations $\theta_j = 2\arcsin(C/\tilde{\lambda}_k)$ are less than 2π, where $\tilde{\lambda}_k$ is an approximation for the eigenvalue λ_j. Hence $C \leqslant \min_j |\lambda_j| = \mathcal{O}(\frac{1}{k})$. For the parameter t_0, the authors define U to be the ideal version of the algorithm in which there is no error in any step. Let $\tilde{U}$ be a version of the algorithm in which only the phase estimation makes error. Since ϵ_H and ϵ_Ψ can be made negligible with relatively few effort, it is sufficient to work with $\tilde{U}$.

Theorem 2 (Error bound)**.** *The error is bounded as*

$$\|\tilde{U} - U\| \leqslant \mathcal{O}(\kappa/t_0). \tag{1}$$

From eq. (1), to guarantee that the error in the final state is $\leqslant \epsilon$, we take $t_0 = \mathcal{O}(\kappa/\epsilon)$.

Now, we analyze the phase estimate. Since $C = \mathcal{O}(1/\kappa)$ and $\lambda \leqslant 1$, then the probability of measuring 1 at *Step 7* is at least $\Omega(1/\kappa^2)$. The authors use amplitude amplification technique [5] to show that $\mathcal{O}(k)$ repetitions of the algorithm are sufficient to measure 1.

Finally, let us give the complexity for the algorithm. Let ϵ be the error at the final state. First, the time to prepare $|\Psi_0\rangle$ up to error ϵ_Ψ is poly $\log(T_H/\epsilon_\Psi)$. Second, the time to simulate e^{iAt} for some $t \leqslant t_0 = \mathcal{O}(\kappa/\epsilon)$ up to error ϵ_H is $\tilde{\mathcal{O}}(\log(n)s^2 t_0) = \tilde{\mathcal{O}}(\log(n)s^2\kappa/\epsilon)$, where the notation $\tilde{\mathcal{O}}(\cdot)$ indicates that $\log^*(n)$ and $9^{\sqrt{\log(s^2 t_0/\epsilon_H)}}$ factors are omitted. Notice that the phase estimation is executed when we simulating the e^{iAt} and the dominant source of error is phase estimation. Furthermore, as we discussed, we need $\mathcal{O}(k)$ repetitions of the algorithm for measuring 1. Putting everything together, the run-time is

$$\tilde{\mathcal{O}}(\kappa \times \log(n)s^2\kappa/\epsilon) = \tilde{\mathcal{O}}(\log(n)s^2\kappa^2/\epsilon),$$

where the notation $\tilde{\mathcal{O}}(\cdot)$ indicates that $\log^*(n)$, $9^{\sqrt{\log(s^2 t_0/\epsilon_H)}}$ and poly $\log(T_H/\epsilon_\Psi)$ factors are omitted.

3.4 Optimality

Concerning optimality, there are two important questions, namely

- classical methods can be improved to reach the complexity of HHL algorithm when only in a summary statistic of the solution, such as $\vec{x}^\dagger M\vec{x}$, is required; and
- HHL algorithm could be improved.

The answer of the first question is no. The authors give a reduction from a general quantum circuit to a matrix inversion problem to obtain that a classical poly$(\log n, \kappa, 1/\epsilon)$-time inversion algorithm could simulate a poly(N)-gate quantum algorithm in poly(N)-time, where $n = \mathcal{O}(2^N\kappa)$. Such a simulation is known to be impossible in the presence of oracles [12]. This reduction gives the answer for the second question is negative too. An improvement to obtain the run-time in polylogarithmic in κ would make $\mathsf{BQP} = \mathsf{PSPACE}$. On the other hand, improving the complexity on poly $\log(1/\epsilon)$ would imply that BQP includes PP.

The authors give their answers by two different points of view. The first one is based on the complexity theory (theorem 3) and the other one is based on oracles (theorem 4). In the framework of of the report, we present only the first point of view in the details, i.e., the one bases on the complexity theory. We recall the *matrix inversion problem* here for convenient.

1. Input: An $\mathcal{O}(1)$-sparse $n \times n$ matrix A. These non-zero entries in a row are returned via an oracle or via a poly$(\log(n))$-time algorithm.

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

The assumptions that A is Hermitian and $\kappa \leqslant \lambda_i \leqslant 1$ for all eigenvalues are needed. The authors take ϵ to be a fixed constant. The algorithm is said to be *relativizing* if the matrix A is specified by an oracle.

The authors also define the *matrix inversion estimation problem*, which is needed for the proof of the third part of theorem 3, as follows. Input: $A, b, M, \epsilon, \kappa, s$ with $\|A\| \leqslant 1$, $\|A^{-1}\| \leqslant \kappa$, A sparse, $|b\rangle = |0\rangle$ 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 $\geqslant 2/3$, where $|x\rangle$ is the unit vector proportional to $A^{-1} |b\rangle$. Moreover, the matrix inversion estimation problem can be solved by quantum computers in time $\tilde{\mathcal{O}}(\log(n)\kappa^2 s^2/\epsilon^3)$.

Theorem 3. *1. If the matrix inversion problem can be solved by a quantum algorithm in run-time* $\kappa^{1-\delta}$.poly$(\log(n))$ *for some* $\delta > 0$*, then* $\mathsf{BQP} = \mathsf{PSPACE}$*.*

2. If there is a classical algorithm with run-time poly$(\kappa, \log(n))$ *for the matrix inversion, then* $\mathsf{BQP} = \mathsf{BPP}$*.*

3. If there is a quantum algorithm for the matrix inversion estimation problem with run-time poly$(\kappa, \log(n), \log(1/\epsilon))$*, then* $\mathsf{PP} = \mathsf{BQP}$*.*

Proof. The important idea is to simulate a quantum circuit by a matrix inversion by using a reduction form a general quantum circuit to a matrix inversion problem. Let $\mathcal{C}$ be a quantum circuit which we want to simulate uses $N = \log(n)$ qubits and T two-qubits gates, namely $U_1, ..., U_T$. The initial state is $|0\rangle^{\otimes N}$ and the answer is obtained by taking measurement in the first qubit of the final state. The authors define a unitary matrix

$$U := \sum_{t=1}^{T} \Big(|t+1\rangle \langle t| \otimes U_t + |t+T+1\rangle \langle t+T| \otimes I + |t+2T+1 \mod 3T\rangle \langle t+2T| \otimes U^\dagger_{3T-t+1} \Big) .$$

We have chosen U such that for $T+1 \leqslant t \leqslant 2T$, then $U^t |1\rangle |\psi\rangle = |t+1\rangle \otimes U_T \cdots U_1 |\psi\rangle$. We define a Hermitian matrix $A := I - Ue^{-\frac{1}{T}}$ which has condition number $\kappa(A) = \mathcal{O}(T)$. We notice that the inverting of A is $A^{-1} = \sum_{k \geqslant U^k e^{-l/T}}$. We take a measurement on the first register. If we obtain $T+1 \leqslant t \leqslant 2T$, then we are left in with the second register $U_T \cdots U_1 |\psi\rangle$ which corresponds to successful computation. This means the matrix inversion problem is BQP-complete.

1. By theorem 1, it suffices to show that $\mathsf{PSPACE} \subseteq \mathsf{BQP}$. Suppose there is a quantum algorithm in run-time $\kappa^{1-\delta}$.poly$(\log(n))$ to solve matrix inversion. Starting from some fix n_0-quibit T_0-gate computation, it is possible to simulate any bigger circuits recursively. Moreover, the run-time is polynomial in n_0. Recall that the TQBF problem is PSPACE-complete. By exhaustive enumeration over all variables, TQBF problem can be solved in time $T \leqslant 2^{2N}/18$. This implies that any problem in PSPACE can be solved in quantum polynomial time. Thus, $\mathsf{PSPACE} \subseteq \mathsf{BQP}$.

2. By theorem 1, it suffices to show that $\mathsf{BQP} \subseteq \mathsf{BPP}$. We simulate a poly$(n)$-time, n-qubit quantum computation as a $\kappa = $ poly(N), $n = 2^N$.poly(N) matrix inversion problem; and apply the classical algorithm to obtain $\mathsf{BQP} \subseteq \mathsf{BPP}$.

3. By theorem 1, it suffices to show that $\mathsf{PP} \subseteq \mathsf{BQP}$. Given a SAT-formula ψ on n variables $z_1, \cdots, x_N$, counting the number of assignments that make the formula true is PP-complete. Given such formula ψ, we can find a quantum circuit that builds the superposition of all 2^N assignments for variables to obtain the state

$$\sum_{(x_1,\cdots,x_N)\in\{0,1\}^N} |x_1,\cdots,x_N\rangle\,|\psi(x_1,\cdots,x_N)\rangle\,.$$

Taking the measurement on the last qubit, the probability of obtaining 1 is equal to the number of satisfying truth assignments divided by 2^N. By using matrix inversion estimation procedure, we can estimate this probability to accuracy $\epsilon = 2^{-2N}$ in time $\text{poly}(\log(1/\epsilon)) = \text{poly}(\log(2^{2N})) = \text{poly}(N)$. This implies that $\mathsf{PP} \subseteq \mathsf{BQP}$.

$\square$

Theorem 4. *1. There is no relativizing quantum algorithm with run-time* $\kappa^{1-\delta}.\text{poly}\log(n)$*.*

2. *There is no relativizing classical algorithm for the matrix inversion problem with the run-time* $n^\alpha 2^{\kappa\beta}$ *unless* $3\alpha + 4\beta \geqslant 1/2$*.*

3. *There is no relativizing quatum algorithm for the matrix inversion problem with run-time* $n^\alpha\text{poly}(\kappa)/\epsilon^\beta$ *unless* $\alpha + \beta \geqslant 1$*.*

To sum up, the part 2 of theorem 3 and the part 2 of theorem 4 give the answer for the first question, that is, the HHL algorithm is really faster than what classical algorithms could achieve. The part 1 of theorem 3 and the part 1 of theorem 4 give HHL algorithm is optimal in term κ. Finally, the third parts of theorem 3 and theorem 4 prove that the HHL algorithm is optimal in term $1/\epsilon$.

3.5 Discussions

The non-Hermitian case:

Suppose $A \in \mathbb{C}^{n\times n}$, we define a new matrix $B \in \mathbb{C}^{(2n)\times(2n)}$ as $B := \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$. Since B is Hermitian, then we can use the algorithm for solving $B\vec{y} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}$ to obtain $\vec{y} = \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix}$. The complexity to solve this general problem is $\tilde{\mathcal{O}}(\log(2n)(2s)^2\kappa^2/\epsilon) = \tilde{\mathcal{O}}(\log(n)s^2\kappa^2/\epsilon)$.

The non-unit vector $\vec{b}$ case:

If the input vector $\vec{b}$ is not a unit vector, we define $\vec{c} = \frac{\vec{b}}{||b||}$ and the matrix B as $B := \frac{A}{||b||}$. Then we can use the algorithm for solving $B\vec{x} = \vec{c}$ with its complexity is $\tilde{\mathcal{O}}(\log(n)s^2\kappa^2/\epsilon)$.

The ill-conditioned case:

A matrix is said to be ill-conditioned if its condition number is very large. Practically, the computation of its inverse, or its solution of a linear system of equations is prone to large numerical errors. A matrix that is not invertible has condition number equal to infinity. To deal with this problem, the authors invert only the part of $|b\rangle$ which is in well-conditioned part of the matrix (the span of eigenspaces corresponds to the eigenvalues $\geqslant 1/\kappa$).

4 Conclusion

Summary:

The authors give a quantum algorithm (HHL algorithm) for linear systems of equations when the input $n \times n$ matrix is s-sparse and well-conditioned. The solution is not exact as we only obtain an approximation solution for this problem. In addition, this solution should be used to estimate an expectation value $\langle x| M |x\rangle$ for some operator M since the time to read all n components, x_i, of $|x\rangle$ is $\mathcal{O}(n)$. The complexity of HHL algorithm is $\tilde{\mathcal{O}}(\log(n)s^2\kappa^2/\epsilon)$, where κ is condition number matrix and ϵ is the total error in the output state $|x\rangle$. Specially, when κ and $1/\epsilon$ equal poly$(\log(n))$, the algorithm achieves an exponential speedup compared to classical algorithms. The complexity of the HHL algorithm in terms of κ and ϵ are proven to be optimal.

Subsequent works:

Ambainis [1] proposed a weakness of HHL algorithm. That is when the condition number κ is taken into account, the run-time pf HHL algorithm is $\mathcal{O}(\kappa^2 \log(n))$. Therefore, if $\kappa = \mathcal{O}(\log^c(n))$, the HHL algorithm stays exponential in run-tim. However, it is more common for a system to have κ equals $\Theta(n)$ or $\Theta(n^c)$. For this reason, in this work, they improve the dependence of HHL algorithm on $1/\kappa$, i.e., improve the run-time from $\mathcal{O}(\kappa^2 \log(n))$ to $\mathcal{O}(\kappa \log^3(\kappa)\log(n))$ by using *variable-time quantum amplitude amplification.*

Recently, Clader, Jacobs and Sprouse [7] (CJS) provided a quantum algorithm with run-time $\tilde{\mathcal{O}}(\kappa s^7 \log(n)/\epsilon^2)$ which is quadratically better in κ than in the HHL algorithm. As we discussed above, the HHL algorithm assume that preparing the generic state $|b\rangle$ is done by other algorithms. CJS provided a general method for efficient preparation of state $|b\rangle$. Moreover, they proposed a deterministic version of the algorithm and a solution to read out all components, x_i, of $|x\rangle$ efficiently.

Recent efforts in small-scale experimental implementation of quantum linear system algorithm can be found in [2, 6].

References

[1] Andris Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In Thomas Wilke Christoph Dürr, editor, *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 636–647, Paris, France, February 2012. LIPIcs.

[2] S. Barz, I. Kassal, M. Ringbauer, Y.O. Lipp, B. Dakic, A. Aspuru-Guzik, and P. Walther. A two-qubit photonic quantum processor and its application to solving systems of linear equations. *Scientific Reports*, 4, Jul 2015.

[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.

[4] Dominic W Berry. High-order quantum algorithm for solving linear differential equations. *Journal of Physics A: Mathematical and Theoretical*, 47(10):105301, 2014.

[5] B. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *Contemporacy Mathematics Series Millenium Volume*, AMS, New York, 2000.

[6] X.-D. Cai, C. Weedbrook, Z.-E. Su, M.-C. Chen, Mile Gu, M.-J. Zhu, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Experimental quantum computing to solve systems of linear equations. *Phys. Rev. Lett.*, 110:230501, Jun 2013.

[7] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. Publisher's note: Preconditioned quantum linear system algorithm [phys. rev. lett. **110** , 250504 (2013)]. *Phys. Rev. Lett.*, 111:049903, Jul 2013.

[8] L. Grover and T. Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. July 2002.

[9] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103(150502), October 2009.

[10] S. Lloyd, M. Mohseni, and P. Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *ArXiv e-prints*, July 2013.

[11] Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994.

[12] Daniel R. Simon. On the power of quantum computation. *SIAM J. Comput.*, 26(5):1474–1483, October 1997.

[13] John Watrous. *Quantum Computational Complexity*, pages 7174–7201. Springer New York, New York, NY, 2009.