Quantum Algorithm for Linear Systems of Equations

Article by:
Aram W. Harrow, Avinatan Hassidim and Seth Lloyd, 2008
Presentation by:
Nir Stiassnie and Shay Kricheli

Quantum Computing, Ben Gurion University

September 2020

Introduction and Outline

- Algorithm for solving linear equations [1]
- Very common in engineering and science

Outline:

- Problem formulation and definitions
- Runtime comparison with classical algorithms, exponential speedup
- Algorithm sketch - concept and details
- Runtime optimality

Problem Formulation

Let:

- $A \in \mathbb{C}^{N \times N}$ be an $N \times N$ Hermitian matrix.
- $\vec{b} \in \mathbb{C}^{N}$ be a N-dimensional vector.

We would like to find a vector $\vec{x}$ satisfying:

$$A\vec{x} = \vec{b}$$

If A is invertible there exists a unique solution which is given by:

$$\vec{x} = A^{-1}\vec{b}$$

- Cases in which A is not invertible will be discussed later on.
- For now we will assume that A is invertible.

Preliminary Definitions

Definition

Given $s \in \mathbb{N}$, a matrix $A \in \mathbb{C}^{N \times N}$ is called s-**sparse** if each row of A contains at most s non-zero entries.

- For any $N \times N$ matrix A: $s = O(N)$.
- In this algorithm, best performance is achieved when: $s \ll N$.
- More specifically, when: s is $poly(\log N)$.
- As in: $\exists k \in \mathbb{N}\ ;\ s = O(\log^k N)$.

Definition

Given an Hermitian matrix $A \in \mathbb{C}^{N \times N}$, the **condition number** of A is given by:

$$\kappa = \frac{|\lambda_{max}|}{|\lambda_{min}|}$$

where λ_{max} and λ_{min} are the maximal and minimal (by moduli) eigenvalues of A respectively.

- κ does not necessarily depend on N.
- In this algorithm, best performance is achieved when: $\kappa \ll N$.
- More specifically, when: κ is $poly(\log N)$.
- κ grows $\rightarrow$ A closer to a singular matrix. Such a matrix is said to be "**ill-conditioned**". If A is not invertible - $\kappa = \infty$.

Classical Algorithms Runtime

- Solving the problem involves inverting A and multiplying the result by $\vec{b}$.
- Theorem 28.1 in [2] states that the matrix multiplication problem is not harder than the matrix inversion problem and theorem 28.2 states vice versa.
- Thus the runtime of the problem is proportional to that of performing matrix inversion.
- Inverting A can be done by performing Gaussian Elimination.

For inverting a general matrix A:

- Gaussian Elimination algorithm - runs in time $O(N^3)$.
- There are minor improvements, up to about $O(N^{2.373})$.
- It is strongly conjured that a tight bound for the matrix multiplication problem given two $N \times N$ matrices is $\Theta(N^2)$.
- Thus the runtime of classical algorithms for matrix inversion is polynomial with high certainty.

Runtime Improvement Attempts

Assuming that A is s-sparse and with condition number κ:

- Conjugate Gradient Descent - Runs in $O(Ns\kappa)$.
- With the assumptions on s and κ, we still get a runtime of $O(N \log^k N)$ for some k. At least polynomial in N.

Even if A is also positive semi-definite:

- The runtime of Conjugate Gradient Descent reduces to $O(Ns\sqrt{\kappa})$.
- We still get a runtime of $O(N \log^r N)$.
- Positive semi-definiteness is an additional assumption the quantum algorithm does not make.

Quantum Algorithm Runtime

- Can a quantum algorithm improve the dependence in N to be better than polynomial time?
- Even when the task is done - just reading out the solution takes $O(N)$ time.
- Let $|x\rangle$ be a n-qubit quantum state (where $n = \log N$) corresponding of the values of $\vec{x}$ up to some error ε.
- In cases where the desired outcome is not $|x\rangle$ itself, but some specific set of functions of $|x\rangle$, the algorithm can perform faster.

Assuming A is s-sparse, for some measurement operator $M \in \mathbb{C}^{N \times N}$ such that $M \geq 0$, as in M is positive semi-definite:

- The expression $\langle x| M |x\rangle$ can be calculated efficiently with error ε - in $poly(\log N, s, \kappa, 1/\varepsilon)$ time.
- More specifically, a runtime of $O(\kappa^2 s^2 \log N/\varepsilon)$, where ε is the error achieved in the output state $|x\rangle$.
- This provides exponential improvement over the best known classical algorithm - in terms of N.
- The total exponential speedup is present where κ, s and $1/\varepsilon$ are $poly(\log N)$.

For example, let $N = 10^{12} \approx 2^{40}$ (and when $\kappa, s, 1/\varepsilon = O(\log(N))$:

- The runtime of the classical algorithm will be at least $\Omega(2^{40})$.
- The runtime of the quantum algorithm will be $poly(\log(2^{40})) = poly(40)$.

Algorithm Idea

Given an Hermitian matrix A:

1. Start with a pre-determined initial state.
2. Performing phase estimation to approximate the the eigenvalues of A.
3. Approximate the inverse of A by inverting its estimated eigenvalues.
4. Use amplitude amplification to maximize the probability for measuring the desired outcome.
5. Perform a measurement using M to estimate $\langle x| M |x\rangle$.

The Non-Hermitian Case

- As the algorithm assumes that A is Hermitian, if A is actually not, then as a preprocess step, we define:

$$D = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}$$

As D is Hermitian, using the algorithm we can now solve the equation:

$$D\tilde{x} = \tilde{b}$$

for $\tilde{x}$, where:

$$\tilde{x} = \begin{bmatrix} 0 \\ \vec{x} \end{bmatrix} \;;\; \tilde{b} = \begin{bmatrix} \vec{b} \\ 0 \end{bmatrix}$$

and then use $\vec{x}$ as explained.

- We want the quantum state $|x\rangle = A^{-1}|b\rangle$.
- We assumed A is invertible. Also Hermitian $\rightarrow$ it is normal.
- Thus its eigenvectors $\{\vec{u_j}\}_{j=1}^N$ corresponding to its eigenvalues $\{\lambda_j\}_{j=1}^N$ consist an orthonormal basis.

Let us denote the representation of $|b\rangle$ in that basis:

$$|b\rangle = \sum_{k=1}^{N} \beta_k |u_k\rangle$$

Claim

The output state can represented as follows:

$$|x\rangle = \sum_{j=1}^{N} \lambda_j^{-1} \beta_j |u_j\rangle$$

Proof

By the spectral theorem $f(A) = \sum_{j=1}^{N} f(\lambda_j) |u_j\rangle \langle u_j|$ and thus: $A^{-1} = \sum_{j=1}^{N} \lambda_j^{-1} |u_j\rangle \langle u_j|$. We have:

$$\begin{aligned} |x\rangle = \quad & A^{-1}|b\rangle = \left(\sum_{j=1}^{N} \lambda_j^{-1} |u_j\rangle \langle u_j| \right) \left(\sum_{k=1}^{N} \beta_k |u_k\rangle \right) \\ = \quad & \sum_{j=1}^{N} \lambda_j^{-1} \beta_j |u_j\rangle \langle u_j|u_j\rangle + \sum_{j=1}^{N} \sum_{\substack{k=1 \\ k \neq j}}^{N} \lambda_j^{-1} \beta_j |u_j\rangle \langle u_j|u_k\rangle \\ = \quad & \sum_{j=1}^{N} \lambda_j^{-1} \beta_j |u_j\rangle \end{aligned}$$

since $\{\vec{u_j}\}_{j=1}^{N}$ consist an orthonormal basis. $\square$

Algorithm Outline

- Input: Oracle access to the rows of an Hermitian matrix A, a method to produce a unit vector $|b\rangle$ and a cutoff value for κ.

1. Represent $\vec{b}$ as a quantum state of the form:

$$|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$$

where: $b_i = \vec{b}[i]$.

2. Next step - produce eigenvalues and eigenvectors of A.
   - Using a simulation of a section of the phase estimation algorithm.
   - Simulate phase estimation $C-U$ section with $U = e^{iAt}$ (which is unitary) via a technique called Hamiltonian Simulation.
   - The Fourier Transform is then applied.
   - This results in a state proportional to:

$$\sum_{j=1}^{N} \beta_j \left|u_j\right\rangle \left|\lambda_j\right\rangle$$

   - We now have produced the eigenvalues in the register.

3. Next step - produce the inverse of the eigenvalues as a scalar. Naive algorithm:
   - Apply conditional rotation on ancilla qubit initialized to $|0\rangle$.
   - Rotate conditioned on the eigenvalues of A - which are all real since it is Hermitian.
   - Let us define the rotation matrix:

$$\tilde{R}_{\lambda_j} = \begin{bmatrix} \sqrt{1 - \frac{1}{\lambda_j^2}} & -\frac{1}{\lambda_j} \\ \frac{1}{\lambda_j} & \sqrt{1 - \frac{1}{\lambda_j^2}} \end{bmatrix}$$

   - But, $\tilde{R}_{\lambda_j}$ is not necessarily unitary.
   - In the case where $|\lambda_j| < 1$, we get $\tilde{R}_{\lambda_j} \tilde{R}^\dagger_{\lambda_j} \neq I$.

- This can be fixed using a normalization constant. Let:

$$R_{\lambda_j} = \begin{bmatrix} \sqrt{1 - \frac{C^2}{\lambda_j^2}} & -\frac{C}{\lambda_j} \\ \frac{C}{\lambda_j} & \sqrt{1 - \frac{C^2}{\lambda_j^2}} \end{bmatrix}$$

- Applying R_{λ_j} to the ancilla qubit we get the form:

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

- Conditioned on measuring 1 in the ancilla qubit, we get a state proportional to:

$$\sum_{j=1}^{N} \beta_j \left|u_j\right\rangle \left|\lambda_j\right\rangle \to C \sum_{j=1}^{N} \lambda_j^{-1} \beta_j \left|u_j\right\rangle \left|\lambda_j\right\rangle$$

  where C is a normalization constant.
- The whole transformation is non-unitary - involves measurement and scaling by a factor $\neq 1$.
- Inverting the eigenvalues is the main challenge solved by the suggested algorithm.

4. Uncompute the $|\lambda_j\rangle$ register, resulting in a state proportional to:

$$\sum_{j=1}^{N} \lambda_j^{-1} \beta_j \, |u_j\rangle = |x\rangle$$

where the equivalence is from the proven claim.

Detailed Algorithm

Let us detail the algorithm operation:

1. Start with an n-qubit register $|initial\rangle$.
2. Produce the state $|b\rangle$. Assume there exists an efficiently implementable unitary operator B such that:

$$B\,|initial\rangle = |b\rangle = \sum_{i=1}^{N} b_i\,|i\rangle$$

possibly along with garbage in ancilla registers. For the discussion of the algorithm, all the errors in B are neglected.

3. Prepare the unitary operator e^{iAt}.

Definition

A Hermitian s-sparse matrix $A \in \mathbb{C}^{N \times N}$, is **efficiently row computable** if it has at most s nonzero entries per row and given a row index $|i\rangle$, the i'th row of A can be computed in time $O(s)$.

- Assume A Hermitian, s-sparse and efficiently row computable.
- Thus we have an oracle access to the rows of A.
- For some time $t \geq 0$ of the evolution, the unitary operator e^{iAt} can be calculated efficiently, as shown in [3].
- In time of approximately $O(\log Ns^2 t)$.
- Combined with the assumption $s \ll N$, this is where the sparsity of A is in fact used.

U_{invert} subroutine

Let us first assume A is well-conditioned.

- Assume the state $|\psi_0\rangle$ can be prepared efficiently, where:

$$|\psi_0\rangle = \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin \pi \left(\frac{\tau + \frac{1}{2}}{T} \right) |\tau\rangle$$

  for some large T such that $T = O(\log N s^2 t)$.
- $|\psi_0\rangle$ are chosen to minimize a loss function of the error.
- Runtime for this operation is $poly(\log(T/\varepsilon))$.

4. Define the subroutine of the algorithm U_{invert}:
   - Initiate a register of zeros noted by L. Prepare $|\psi_0\rangle$ on register L and adjoin it to $|b\rangle$, to result in:

     $$|\psi_0\rangle \otimes |b\rangle$$

     For phase estimation, apply the next two steps:
   - Apply conditional Hamiltonian evolution on $|\psi_0\rangle \otimes |b\rangle$ with:

     $$\sum_{\tau=0}^{T-1} |\tau\rangle \langle\tau| \otimes e^{iAt_0\tau/T}$$

     for some chosen t_0 such that $t_0 = O(\kappa/\varepsilon)$.
   - Apply the Fourier transform to register L.

After the Fourier transform, we end up with the state:

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

where $|k\rangle$ are the Fourier basis states and $|\alpha_{k,j}|$ is close to 1 if and only if $\lambda_j \approx \frac{2\pi k}{t_0}$. Let $\tilde{\lambda_k} = \frac{2\pi k}{t_0}$. We can relabel the state to be:

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

To get $\tilde{\lambda_k}^{-1}$, apply the next non-unitary operation:

- Adjoin a register S in the state:

$$|h(\tilde{\lambda_k})\rangle = \sqrt{1 - f(\tilde{\lambda_k})^2}\,|0\rangle + f(\tilde{\lambda_k})\,|1\rangle$$

  where:

  - '0' indicates that the desired matrix inversion hasn't taken place yet.
  - '1' indicates that it has.
  - f is called a filter function - used to produce the inverse of the eigenvalues. They adhere certain conditions that we will discuss later.
  - This is a generalization of what we saw earlier.

We end up with the following state:

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

- To uncompute the $\tilde{\lambda_k}$ register, reverse the first three steps to undo phase estimation.
- If the phase estimation was perfect, we would have $\alpha_{k,j}=1$ if $\tilde{\lambda_k}=\lambda_j$ and 0 otherwise.
- Assuming this is the case, we get the state:

$$\sum_{j=1}^{N}\beta_j\left|u_j\right\rangle\left(\sqrt{1-f(\tilde{\lambda_k})^2}\left|0\right\rangle+f(\tilde{\lambda_k})\left|1\right\rangle\right)$$

Main Loop - Amplitude Amplification

5. Given the initial state $|\phi_0\rangle = U_{invert}B\,|initial\rangle$, apply the following repeatedly:

$$U_{invert}BR_{init}B^{\dagger}U_{invert}^{\dagger}R_{succ}$$

- where:
  - $R_{succ} = I - 2\,|1\rangle\langle 1|$ (reflection over $|1\rangle^{\perp}$)
  - $R_{init} = I - 2\,|initial\rangle\langle initial|$ (reflection over $|initial\rangle^{\perp}$)
- Measure register L at the end of the loop until $|1\rangle$ is measured

Claim

Given p the probability to measure $|1\rangle$ in $|\phi_0\rangle$ - the amplitude amplification procedure makes $O\left(\frac{1}{\sqrt{p}}\right)$ repetitions.

Proof

For some small θ, the initial state $|\phi_0\rangle$ can be represented as:

$$|\phi_0\rangle = \cos(\theta)\,|0\rangle + \sin(\theta)\,|1\rangle$$

Thus $p = \sin^2(\theta)$. Since θ is small and thus $p \approx \theta^2$. As with amplitude amplification, each repetition increases the angle by 2θ (later elaborated). Thus after n repetitions the state becomes:

$$|\phi_{n+1}\rangle = \cos((2n+1)\theta)\,|0\rangle + \sin((2n+1)\theta)\,|1\rangle$$

The amplitude is maxed when the coefficient of $|1\rangle$ is close to 1, as in:

$$(2n+1)\theta \approx \frac{\pi}{2} \to n \approx \frac{\pi}{4\theta} = \frac{\pi}{4\sqrt{p}} \;\square$$

6. After measuring $|1\rangle$, the state is proportional to :

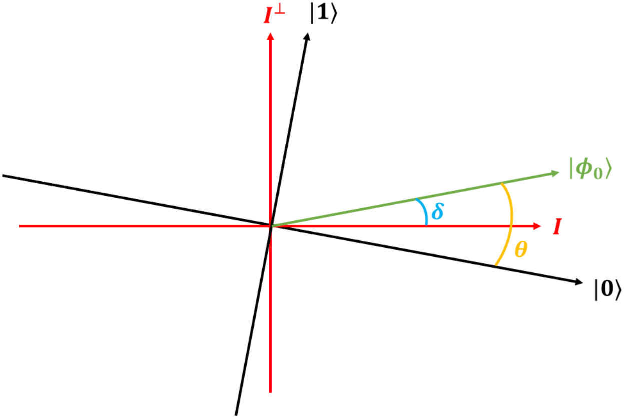
$$\sum_{j=1}^{N} \lambda_j^{-1} \beta_j \left| u_j \right\rangle = \left| x \right\rangle$$

7. Perform a measurement with respect to $\{M, I - M\}$ as the POVM to achieve an estimate of $\langle x | M | x \rangle$ up to error ε.

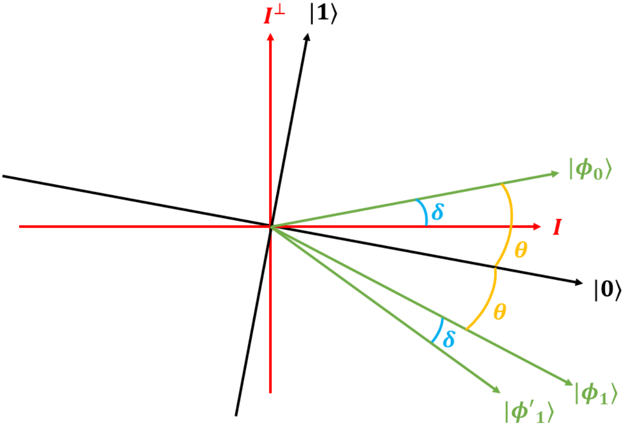
Amplitude Amplification Illustration

- We assumed the initial state is $|\phi_0\rangle = \cos(\theta)|0\rangle + \sin(\theta)|1\rangle$.
- Thus the initial angle relative to the $|0\rangle$ axis is θ.
- Let us denote by δ the angle incurred by the transformation $U_{invert}B$ and assume w.l.o.g the rotation is counterclockwise.
- Thus the rotation by $B^\dagger U_{invert}^\dagger$ is a clockwise rotation by δ.

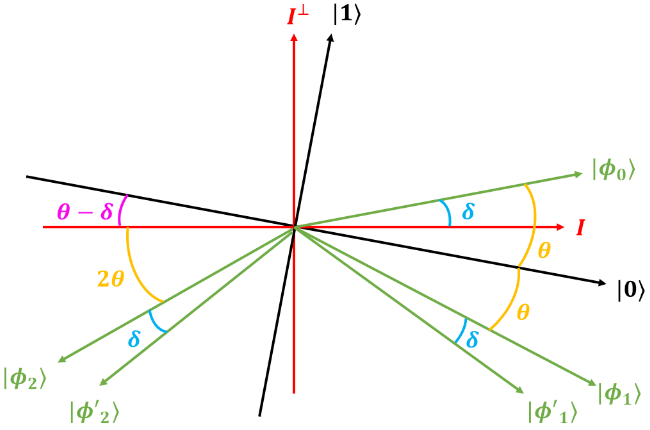
The initial state $|\phi_0\rangle = U_{invert}B\,|initial\rangle$ corresponds to a counterclockwise rotation of δ from $|initial\rangle$:



After reflection by $|1\rangle^{\perp} = |0\rangle$ and clockwise rotation by δ:



After reflection by $|initial\rangle^{\perp}$ and counterclockwise rotation by δ:



Total angle by $|0\rangle$ is $\theta - \delta + 2\theta + \delta = 3\theta$.

Ill-Conditioned Case

The algorithm can also handle ill-conditioned matrices.

Definition

Given a eigenvalue λ of a matrix $A \in \mathbb{C}^{N \times N}$, the **eigenspace** of λ is defined as:

$$E_\lambda = \{v \mid (\lambda I - A)v = 0\}$$

Given a Hermitian matrix $A \in \mathbb{C}^{N \times N}$ with condition number κ, the **well-conditioned part of** A is defined as:

$$span\left(\bigcup_{\lambda \geq 1/\kappa} E_\lambda\right)$$

Ill-conditioned part of A is symmetrically defined for $\lambda < 1/\kappa$.

- To handle ill-conditioned matrices, the algorithm inverts only the part of $|b\rangle$ which is in the well-conditioned part of A.
- Formally, instead of transforming $|b\rangle = \sum_j \beta_j |u_j\rangle$ to $|x\rangle = \sum_j \lambda_j^{-1}\beta_j |u_j\rangle$, transform into a state close to:

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

  in time $O(\kappa^2)$.
- The last qubit is a flag that enables to estimate the size of the ill-conditioned part.
- Handles cases where A is not invertible and produces the projection of $|b\rangle$ on the well-conditioned part of A.

Ill-Conditioned U_{invert}

- In the ill-conditioned case, U_{invert} is changed.
- The S register in step 4 is altered to:

$$|h(\tilde{\lambda_k})\rangle = \sqrt{1 - f(\tilde{\lambda_k})^2 - g(\tilde{\lambda_k})^2}\,|0\rangle + f(\tilde{\lambda_k})\,|1\rangle + g(\tilde{\lambda_k})\,|2\rangle$$

- g is also a filter function, same as f.
- '2' indicates part of $|b\rangle$ is in the ill-conditioned subspace of A.

Filter Functions

- f and g defined earlier are filter functions.
- They set the amplitudes of the basis states of $|h(\lambda)\rangle$.
- Setting $\kappa' = 2\kappa$, we define a new range where f and g return values in between their maximal and minimal values.
- f and g adhere the following conditions for some constant $K > 1$:
  - $f(\lambda) = \frac{1}{K\lambda}$ for $\lambda \geq \frac{1}{\kappa}$.
  - $f^2(\lambda) + g^2(\lambda) \leq 1$.
  - $g(\lambda) = \frac{1}{K}$ for $\lambda \leq \frac{1}{\kappa'}$.

Example for a f and g for $K = 2$:

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

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

Optimality

- The quantum algorithm run-time is $O(\kappa^2 s^2 \log N/\varepsilon)$.
- Article shows optimality in κ and $1/\varepsilon$.
- It also shows no classical algorithm can run in this time.
- We will discuss optimality in κ and with relation to classical algorithms.

Optimality In κ

- The runtime dependency in κ is polynomial.
- The dependency in κ **can not** be improved to be polylogaritmic.
- Furthermore, it **can not** be improved to $\kappa^{1-\delta}$ for $\delta > 0$.
- The proof of this statement is based on arguments from complexity theory.

Definition

Let MI denote the set of all algorithms that solve matrix inversion.

Theorem

Let $MIQ \subset MI$ be the set of all quantum algorithms that solve matrix inversion for a $N \times N$ matrix with condition number κ. Then if there exists $\mathcal{A} \in MIQ$ that has a error of ε and runs in $\kappa^{1-\delta} poly(\log N, 1/\varepsilon)$ time, for some $\delta > 0$ then **BQP** $=$ **PSPACE**.

- It is highly unlikely that **BQP** $=$ **PSPACE** - so this results in optimality.
- The proof is based on a reduction from a general n-qubit quantum circuit with T gates to a matrix inversion problem with:
  - $N = O(T2^n)$
  - $\kappa = O(T)$

Proof Outline

- TQBF (quantified SAT problem) problem - known to be **PSPACE**-complete, solvable for input of size n with n qubits and $T = \Theta(2^{2n})$ gates.
- Using the reduction, we get a matrix inversion problem equivalent to solving TQBF, where $N = O(2^{3n})$. For sufficiently large n, the error increases, specifically $\varepsilon \geq 1/\log(n)$ with runtime:

$$\kappa^{1-\delta}\left(\frac{\log N}{\varepsilon}\right)^{c_1} \leq T^{1-\delta}\left(\frac{3n}{\varepsilon}\right)^{c_1} \leq T^{1-\delta}c_2(n\log n)^{c1}$$

for some $c_1, c_2 > 0$.

- Given a constant $m = \frac{2}{\delta}\frac{\log(2n)}{\log(\log n)}$
- Iterating the reduction for $l \leq m$ steps repeatedly, for each step i:

$$\begin{aligned} T_{i+1} &= T_i^{1-\delta} c_2 (n \log n)^{c1} \\ n_{i+1} &= n_i + \log(18\ T_i) \end{aligned}$$

- $l = min\{m, i\}$, where i is the first iteration $T_{i+1} > T_i^{1-\delta/2}$
- Setting $n_0 = n$, this results in T_l is $poly(n_0)$.
- n_l is shown to be also $poly(n_0)$.
- Thus a **PSPACE** computation can be solved in quantum polynomial time.

Optimality Relating to Classical Algorithms

Theorem

Let $MIC \subset MI$ be the set of all classical algorithms that solve matrix inversion for a $N \times N$ matrix with condition number κ. Then if there exists $\mathcal{A} \in MIC$ that runs in $poly(\kappa, \log N)$ time, for some $\delta > 0$ then **BPP** $=$ **BQP**.

Proof outline:

- A problem in **BQP** with n qubits and $T = poly(n)$ gates is reduced to a inverting a matrix with with $\kappa = poly(n)$, $N = 2^n poly(n)$.
- Assuming the specified classical algorithm exists, we get a $poly(n)$ runtime, thus **BPP** $=$ **BQP**.

Open Questions

- Can we find another quantum algorithm for solving linear systems of equations/matrix inversion using the following formula?
$$A^{-1} = \frac{1}{\det(A)} adj(A)$$
- Can we find similar quantum algorithm providing exponential speedup for other matrix operations such as determinants, adjacent matrix and such?
- Can we improve the dependency in N to be better than $\log N$?
- Can we find an efficient quantum algorithm for solving linear systems of equations for non-sparse matrices?

References

[1] Harrow, Aram W., Avinatan Hassidim, and Seth Lloyd. *Quantum Algorithm for Linear Systems of Equations*. Physical Review Letters 103.15 (2009): n. pag. Crossref. Web. arXiv:0811.3171.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms, Third Edition*. MIT Electrical Engineering and Computer Science.

[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.