Solving linear systems of equations on a quantum computer

arXiv:0811.3171

PRL **15**:150502 (2009)

Aram Harrow[1] Avinatan Hassidim[2] Seth Lloyd[2]

[1]University of Bristol

[2]MIT

Rutgers

April 9, 2010

Outline

- The problem.
- Classical solutions.
- Our quantum solution.
- How it works.
- BQP-completeness / (near-)optimality
- Related work / extensions / applications.

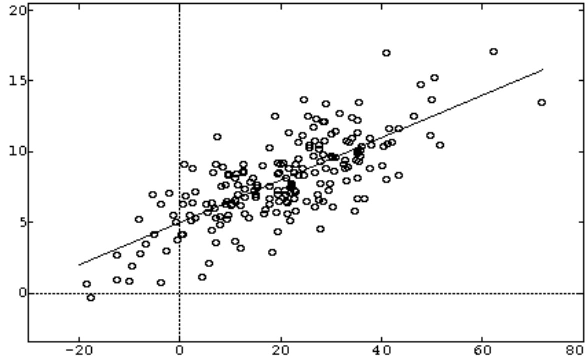
Goal: solving linear systems of equations

- We are given A, a Hermitian $N \times N$ matrix.
- $\vec{b} \in \mathbb{C}^N$ is also given as input.
- We want to (approximately) find $\vec{x} \in \mathbb{C}^N$ such that $A\vec{x} = \vec{b}$.
- If A is not Hermitian or square, we can use $\begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$. Why?

  Because
  $$\begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}.$$
- Some weaker goals are to estimate $\vec{x}^\dagger M \vec{x}$ (for some matrix M) or sample from the probability distribution $\Pr[i] \propto |x_i|^2$.

Application: linear regression

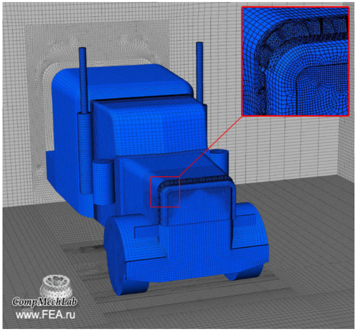
Goal: find the best fit line, low-degree polynomial, etc.



Technically we choose $\vec{x}$ to minimise $\|A\vec{x} - \vec{b}\|^2$.

Application: partial differential equations

Approximate a continuous function with a finite element model.



$$\frac{\partial}{\partial t} \longrightarrow \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

$$\frac{\partial^2}{\partial t^2} \longrightarrow \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}$$

A is a discretised PDE, $\vec{b}$ specifies boundary conditions.
$\vec{x}$ gives solution to boundary-value problem.

Classical algorithms

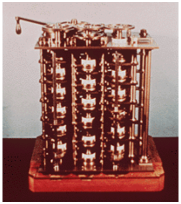
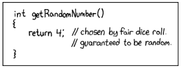
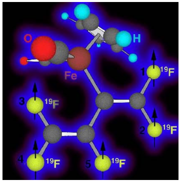
- The LU decomposition finds $\vec{x}$ in time $O(N^{2.376}\,\mathrm{poly}(\log(\kappa/\epsilon)))$.
  - Here "2.376" is the matrix-multiplication exponent.
    (By contrast, Gaussian elimination takes time $O(N^3)$.)
  - ϵ is a bound on error in $\vec{x}$.
  - κ is the condition number.

$$\kappa = \|A\| \cdot \|A^{-1}\| = \frac{\sigma_1(A)}{\sigma_N(A)}$$

  Here $\sigma_i(A)$ is the i^{th} singular value and $\|A\| = \sigma_1(A)$.
  κ measures how hard A is to invert, or equivalently, how sensitively A^{-1} depends on changes in A.

- Iterative methods (e.g. conjugate gradient) require $O(\sqrt{\kappa}\log(1/\epsilon))$ matrix-vector multiplications.
  - If A is s-sparse (i.e. has $\leq s$ nonzero entries per row) then the total time is $O(Ns\sqrt{\kappa}\log(1/\epsilon))$.
  - $|\mathrm{support}(\vec{b})| \cdot (s/\epsilon)^{O(\sqrt{\kappa})} \cdot \mathrm{poly}(\log(N))$ is also possible.

Quantum computing review

	Classical 	Randomised 	Quantum 
basic unit of information	bit: $b \in \{0,1\}$	distribution $p \in \mathbb{R}^2$ $p_0 + p_1 = 1$	qubit $\lvert\psi\rangle \in \mathbb{C}^2$ $\lvert\psi_0\rvert^2 + \lvert\psi_1\rvert^2 = 1$
n bits	2^n states	2^n dimensions	2^n dimensions
basic unit of computation	NAND, XOR, etc.	stochastic matrices	unitary matrices
poly-time	**P**	**BPP**	**BQP**
measurement	no problem	Bayes' rule	collapses state

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Our results

- Quantum Algorithm. Suppose that
  - $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$ is a unit vector that can be prepared in time T_B;
  - A is s-sparse, efficiently row-computable and $\kappa^{-1} I \leq |A| \leq I$
  - $|x'\rangle = A^{-1}|b\rangle$ and $|x\rangle = \frac{|x'\rangle}{\| |x'\rangle \|}$.

  Then our (quantum) algorithm produces $|x\rangle$ and $\langle x'|x'\rangle$, both up to error ϵ, in time

  $$\tilde{O}(\kappa T_B + \log(N) s^4 \kappa^2 / \epsilon).$$

  Reminder: classical algorithms output the entire vector $\vec{x}$ in time $\tilde{O}(\min(N^{2.376}, Ns\sqrt{\kappa}, (s/\epsilon)^{O(\sqrt{\kappa})}))$. This is exponentially slower when $s = O(1)$ and $\kappa = \operatorname{poly}\log(N)$.
- Optimality. Given plausible complexity-theoretic assumptions, these run-times (both quantum and classical) cannot be improved by much. Argument is based on BQP-hardness of the matrix inversion problem.

Algorithm idea: diagonal case

- Suppose A is diagonal:

$$A = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & \\ \vdots & & \ddots & \\ 0 & & & \lambda_N \end{pmatrix}$$

  Then our task is called filtering.

- Our desired transform is the non-unitary operation: $|i\rangle \rightarrow \lambda_i^{-1} |i\rangle$.
- This can be achieved probabilistically by
  1. Choosing c such that $c|\lambda^{-1}| \leq 1$.
  2. Mapping $|i\rangle \rightarrow (\sqrt{1 - c^2|\lambda_i|^{-2}}\,|0\rangle + c\lambda_i^{-1}\,|1\rangle) \otimes |i\rangle$
  3. Measuring the first qubit and hoping we get outcome "1".

  The resulting state is proportional to $A^{-1}|b\rangle$.

Algorithm idea: general case

- If we could work within the eigenbasis of A, then we could use the diagonal algorithm.
- Finding the eigenbasis of A is done with two primitives:
  - Hamiltonian simulation. We can apply e^{iAt} in time $\tilde{O}(ts^4 \log(N))$. (Uses fancy versions of $e^{i(A_1+A_2)\epsilon} \approx e^{iA_1\epsilon}e^{iA_2\epsilon}$.)[1]
  - Phase estimation. Applying $e^{i\lambda t}$ for a carefully chosen superposition[2] of times from 0 to t_0 can be used to produce $\tilde{\lambda} \approx \lambda \pm O(1/t_0)$.
- Phase estimation on e^{iAt} automatically resolves $|b\rangle$ into the eigenbasis of A by (approximately) measuring λ.
- Doing this coherently can (approximately) map $|b\rangle$ to

$$|0\rangle \otimes \sqrt{I - c^2A^{-2}}\,|b\rangle + |1\rangle \otimes cA^{-1}\,|b\rangle\,,$$

  where c is chosen so that $\|cA^{-1}\| \leq 1$.
- Measure the first qubit. Upon outcome "1" we are left with $|x\rangle$.

[1]D.W. Berry, G. Ahokas, R. Cleve and B.C. Sanders. Efficient Quantum algorithms for sparse Hamiltonians. *CMP 2007*, quant-ph/0508139.

[2]V. Buzek, R. Derka and S. Massar. Optimal quantum clocks. *PRL 1999*.

Analysis of the algorithm

- The Hamiltonian simulation produces negligible error. (Error ϵ incurs overhead of $\exp(O(\sqrt{\log(1/\epsilon)})) = \epsilon^{-o(1)}$.) Recall that it takes time $\tilde{O}((\log N)s^4 t_0)$.
- Phase estimation produces error of $O(1/t_0)$ with tail probability dying off fast enough to not bother us.
- An additive error of $1/t_0$ in λ translates into an error in λ^{-1} of $\lambda^{-2}/t_0 \leq \kappa^2/t_0$. Thus, we can take $t_0 \sim \kappa^2/\epsilon$.
- We can take $C = 1/2\kappa$ to guarantee that $\|CA^{-1}\| \leq 1/2$. ($C = 1/\kappa$ should work, but the analysis is more painful.)
- Thus post-selection succeeds with probability at least $O(1/\kappa^2)$ and blows up error by at most $O(\kappa)$. With enough algebra, the run-time magically stays at $O(\kappa^2/\epsilon)$.
- Our best lower bound for the run-time is κ.

How large is κ?

- For many practical problems, it is N^c, in which case our speedup ranges from polynomial to not a speedup.
- There are many classical techniques used to reduce condition number to $N^{o(1)}$ or even poly $\log(N)$, e.g. multi-scale methods and preconditioners [D. Spielman and S.-H. Teng. "Nearly-Linear Time Algorithms for Preconditioning and Solving Symmetric, Diagonally Dominant Linear Systems" arXiv:cs/0607105]. However, their applicability to quantum algorithms is not obvious.
- For random Hermitian matrices, $\kappa = N^{O(1)}$. But if A is chosen independently of $\vec{x}$, then $\vec{b}$ will have low overlap with the small eigenvalues, and we can find $|x\rangle$ up to error ϵ in time poly$(\log(N)/\epsilon)$.
- Finite-element models have extremely wide application. If we take lattice spacing h in d dimensions, then we have $N = h^d$, $s \sim d$ and $\kappa \sim dh$. For best results, fix h and let d grow.

Q-sampling $|x\rangle$ vs. computing $\vec{x}$

Types of solutions: roughly from strongest to weakest

1. Output $\vec{x} = (x_1, \ldots, x_N)$. *Classical algorithms*
2. Produce $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$. *Our algorithm*
3. Sample i according to $p_i \sim |\langle i|x\rangle|^2$.
4. Estimate $\langle x| M |x\rangle$ for some (perhaps diagonal) matrix M.

Compare with classical Monte Carlo algorithms

The old-fashioned way to get an exponential speed-up.

- They work with a sample drawn from $\vec{p} = (p_1, \ldots, p_N)$.
- If A is stochastic and sparse then $\vec{p} \mapsto A\vec{p}$ is efficient.
- If $-1 \leq m_1, \ldots, m_N \leq 1$, then $\sum_{i=1}^{N} m_i p_i$ can be estimated to error ϵ using $O(1/\epsilon^2)$ samples.

Is matrix inversion easier if we only need to estimate $\vec{x}^\dagger M \vec{x}$?

BQP-hardness of matrix inversion

Consider a quantum circuit on n qubits that starts in the state $|0\rangle^{\otimes n}$, applies two-qubit gates $U_1, \ldots, U_T$ and then measures the first qubit.

Theorem

Estimating the acceptance probability of this circuit reduces to estimating $\langle x| M |x\rangle$ where M is diagonal, $A\vec{x} = \vec{b}$, $\vec{b} = |0\rangle$, A has dimension $N = O(T2^n)$ and $\kappa = O(T)$.

Corollary

- A classical poly$(\log(N), \kappa)$ algorithm for estimating $\langle x| M |x\rangle$ to constant accuracy would imply BPP=BQP (i.e. randomized algorithms are as strong as quantum algorithms).
- If we allow only black-box access to A, then no quantum algorithm can run in time $\kappa^{1-\delta} \cdot \text{poly}\log(N)$ or $\text{poly}(\kappa) \cdot (N/\epsilon)^{o(1)}$.

Proof of BQP-hardness

An idea that almost works

- Our quantum circuit is $U_T \cdots U_1$.
- On the space $\mathbb{C}^T \otimes \mathbb{C}^{2^n}$ define

$$V = \sum_{t=1}^{T} |t+1 \pmod{T}\rangle \langle t| \otimes U_t. \qquad \text{is unitary}$$

$$A = I - e^{-\frac{1}{T}} V \qquad \text{has } \kappa \leq 2T$$

- Expand

$$A^{-1} = \sum_{k=0}^{\infty} e^{-\frac{k}{T}} V^k$$

So that $\kappa^{-1} A^{-1} |1\rangle |\psi\rangle$ has $\Omega(1/T)$ overlap with

$$V^T |1\rangle |\psi\rangle = |1\rangle U_T \cdots U_1 |\psi\rangle .$$

But undesirable terms contribute too.

Proof of BQP-hardness

The correct version

- Define
$$U_{T+1} = \ldots = U_{2T} = I^{\otimes n}$$
$$U_{2T+1} = U_T^\dagger, \ldots, U_{3T} = U_1^\dagger$$
so that $U_{3T} \ldots U_1 = I^{\otimes n}$ and $U_t \ldots U_1 = U_T \ldots U_1$ whenever $T \leq t < 2T$.
- Now define (on the space $\mathbb{C}^{3T} \otimes \mathbb{C}^{2^n}$) the operators
$$V = \sum_{t=1}^{3T} |t+1 \pmod{3T}\rangle \langle t| \otimes U_t$$
$$A = I - e^{-\frac{1}{T}} V$$
- This time $\kappa^{-1} A^{-1} |1\rangle |\psi\rangle$ has $\Omega(1)$ overlap with successful computations (i.e. $|t\rangle \otimes U_T \ldots U_1 |\psi\rangle$ for $T \leq t < 2T$) and there is no extra error from wrap-around.

Related work

- [L. Sheridan, D. Maslov and M. Mosca. Approximating Fractional Time Quantum Evolution. 0810.3843] show how access to U can be used to simulate U^t for non-integer t.
- [S.K. Leyton and T.J. Osborne. A quantum algorithm to solve nonlinear differential equations. 0812.4423] requires time $\text{poly}\log(\text{number of variables}) \cdot \exp(\text{integration time})$.
- [Szkopek et al., Eigenvalue Estimation of Differential Operators with a Quantum Algorithm, quant-ph/0408137] has similar scaling, and also resembles our application to finite-element methods.
- [S. P. Jordan and P. Wocjan. Efficient quantum circuits for arbitrary sparse unitaries. 0904.2211] is also based on Hamiltonian simulation.
- [D. Janzing and P. Wocjan. Estimating diagonal entries of powers of sparse symmetric matrices is BQP-complete. quant-ph/0606229] is similar to our BQP-hardness result.

Extensions: known and unknown

(Mostly unknown)

- If A is ill-conditioned, we can choose κ arbitrarily, invert the part with eigenvalues $\gg 1/\kappa$ and flag the bad part with eigenvalues $\ll 1/\kappa$ (with some gray area around the $1/\kappa$ threshold).
- If $\|A\| \gg 1$, then we should be able to rescale A and disregard large eigenvalues of A that contribute very little to A^{-1}.
  This raises a Hamiltonian simulation problem of independent interest: how costly is it to simulate a high-energy theory on low-energy states?
- B is a preconditioner if $\kappa(AB) \ll \kappa(A)$. If B is sparse, then BA is as well, and we can apply $(BA)^{-1}$ to $B|b\rangle$. Preconditioners are crucial to practical (classical) iterative methods and we would like to make use of them with our algorithm.
- More applications, please! Candidates are deconvolution, solving elliptical PDE's and speeding up linear programming.