Title: Quantum algorithm for solving linear systems of equations

Date: May 04, 2009 04:00 PM

URL: http://pirsa.org/08050061

Abstract: Solving linear systems of equations is a common problem that arises both on its own and as a subroutine in more complex problems: given a matrix A and a vector b, find a vector x such that Ax=b. Often, one does not need to know the solution x itself, but rather an approximation of the expectation value of some operator associated with x, e.g., x'Mx for some matrix M. In this case, when A is sparse and well-conditioned, with largest dimension N, the best known classical algorithms can find x and estimate x'Mx in O(N * poly(log(N))) time.
In this talk I'll describe a quantum algorithm for solving linear sets of equations that runs in poly(log N) time, an exponential improvement over the best classical algorithm.

This talk is based on my paper arXiv:0811.3171v2, which was written with Avinatan Hassidim and Seth Lloyd.

A Quantum algorithm for solving $A\vec{x} = \vec{b}$

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

[1] University of Bristol

[2] MIT

Perimeter Institute seminar
4 May, 2009

Outline

- The problem.
- Classical solutions.
- Our quantum solution.
- How it works.
- Why it's (not so far from) optimal.
- 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 $A\vec{x} = \vec{b}$.

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
$$\begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}.$$

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 $\Pr[i] \propto |x_i|^2$.

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 $\Pr[i] \propto |x_i|^2$.
- This problem was introduced in middle school, and has applications throughout high school, college, grad school and even work.

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 $\vec{x}$.

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.

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.

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 $O(Ns\sqrt{\kappa}\log(1/\epsilon))$.

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.

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;

Aram Harrow

University of

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 $\kappa^{-1} I \leq |A| \leq I$

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 $|x\rangle = \frac{|x'\rangle}{\sqrt{\langle x'|x'\rangle}}$.

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 $|x\rangle = \frac{|x'\rangle}{\sqrt{\langle x'|x'\rangle}}$.

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

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

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 $|x\rangle = \frac{|x'\rangle}{\sqrt{\langle x'|x'\rangle}}$.

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

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

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 $\kappa = \text{poly}\log(N)$.

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 $|x\rangle = \frac{|x'\rangle}{\sqrt{\langle x'|x'\rangle}}$.

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

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

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 $\kappa = \operatorname{poly}\log(N)$.

Aram Harrow

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 $|x\rangle = \frac{|x'\rangle}{\sqrt{\langle x'|x'\rangle}}$.

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

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

  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

Algorithm idea

- Based on two key primitives:
  - Hamiltonian simulation. Trotter techniques[1] can be used to simulate e^{iAt} in time $\tilde{O}(ts^2 \log(N))$.

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

Algorithm idea

- Based on two key primitives:
  - Hamiltonian simulation. Trotter techniques[1] can be used to simulate e^{iAt} in time $\tilde{O}(ts^2 \log(N))$.
  - 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 $\tilde{\lambda} \approx \lambda \pm O(1/t_0)$.

[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*, quant-ph/9808042.

Algorithm idea

- Based on two key primitives:
  - Hamiltonian simulation. Trotter techniques[1] can be used to simulate e^{iAt} in time $\tilde{O}(ts^2 \log(N))$.
  - 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.

[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*, quant-ph/9808042.

Algorithm idea

- Based on two key primitives:
  - Hamiltonian simulation. Trotter techniques[1] can be used to simulate e^{iAt} in time $\tilde{O}(ts^2 \log(N))$.
  - 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

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

where c is chosen so that $\|cA^{-1}\| \leq 1$.

---

[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*, quant-ph/9808042.

Algorithm idea

- Based on two key primitives:
  - Hamiltonian simulation. Trotter techniques[1] can be used to simulate e^{iAt} in time $\tilde{O}(ts^2 \log(N))$.
  - 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

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

  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*, quant-ph/9808042.

Algorithm details

Let $|b\rangle = \sum_\lambda b_\lambda |u_\lambda\rangle$.

Algorithm details

Let $|b\rangle = \sum_\lambda b_\lambda |u_\lambda\rangle$.

1. Prepare control register in superposition of $|t\rangle$ over $0 \leq t \leq t_0$.

Algorithm details

Let $|b\rangle = \sum_\lambda b_\lambda |u_\lambda\rangle$.

1. Prepare control register in superposition of $|t\rangle$ over $0 \le t \le t_0$.
2. Use Hamiltonian simulation to apply $\sum_t |t\rangle\langle t| \otimes e^{iAt}$.

Aram Harrow

University of

Algorithm details

Let $|b\rangle = \sum_\lambda b_\lambda |u_\lambda\rangle$.

1. Prepare control register in superposition of $|t\rangle$ over $0 \le t \le t_0$.
2. Use Hamiltonian simulation to apply $\sum_t |t\rangle\langle t| \otimes e^{iAt}$.
3. Fourier transform first register, yielding

$$\sum_{\lambda,\tilde{\lambda}} \alpha_{\lambda,\tilde{\lambda}} \left|\tilde{\lambda}\right\rangle \otimes b_\lambda |u_\lambda\rangle ,$$

with $|\alpha_{\lambda,\tilde{\lambda}}|$ small unless $\tilde{\lambda} \approx \lambda$.

Algorithm details

Let $|b\rangle = \sum_\lambda b_\lambda |u_\lambda\rangle$.

1. Prepare control register in superposition of $|t\rangle$ over $0 \leq t \leq t_0$.
2. Use Hamiltonian simulation to apply $\sum_t |t\rangle\langle t| \otimes e^{iAt}$.
3. Fourier transform first register, yielding

$$\sum_{\lambda,\tilde{\lambda}} \alpha_{\lambda,\tilde{\lambda}} \left|\tilde{\lambda}\right\rangle \otimes b_\lambda |u_\lambda\rangle ,$$

   with $|\alpha_{\lambda,\tilde{\lambda}}|$ small unless $\tilde{\lambda} \approx \lambda$.

4. Conditioned on $\tilde{\lambda}$, adjoin state

$$\sqrt{1 - C^2\tilde{\lambda}^{-2}}\,|0\rangle + C\tilde{\lambda}^{-1}\,|1\rangle .$$

Algorithm details

Let $|b\rangle = \sum_\lambda b_\lambda |u_\lambda\rangle$.

1. Prepare control register in superposition of $|t\rangle$ over $0 \le t \le t_0$.
2. Use Hamiltonian simulation to apply $\sum_t |t\rangle\langle t| \otimes e^{iAt}$.
3. Fourier transform first register, yielding

$$\sum_{\lambda,\tilde{\lambda}} \alpha_{\lambda,\tilde{\lambda}} \left|\tilde{\lambda}\right\rangle \otimes b_\lambda |u_\lambda\rangle ,$$

   with $|\alpha_{\lambda,\tilde{\lambda}}|$ small unless $\tilde{\lambda} \approx \lambda$.

4. Conditioned on $\tilde{\lambda}$, adjoin state

$$\sqrt{1 - C^2\tilde{\lambda}^{-2}}\,|0\rangle + C\tilde{\lambda}^{-1}\,|1\rangle .$$

5. Undo steps 1-3

Algorithm details

Let $|b\rangle = \sum_\lambda b_\lambda |u_\lambda\rangle$.

1. Prepare control register in superposition of $|t\rangle$ over $0 \le t \le t_0$.
2. Use Hamiltonian simulation to apply $\sum_t |t\rangle\langle t| \otimes e^{iAt}$.
3. Fourier transform first register, yielding

$$\sum_{\lambda,\tilde{\lambda}} \alpha_{\lambda,\tilde{\lambda}} \left|\tilde{\lambda}\right\rangle \otimes b_\lambda |u_\lambda\rangle ,$$

   with $|\alpha_{\lambda,\tilde{\lambda}}|$ small unless $\tilde{\lambda} \approx \lambda$.
4. Conditioned on $\tilde{\lambda}$, adjoin state

$$\sqrt{1 - C^2\tilde{\lambda}^{-2}}\,|0\rangle + C\tilde{\lambda}^{-1}\,|1\rangle .$$

5. Undo steps 1-3
6. Measure ancilla qubit and start over if outcome isn't 1. (Technically, use amplitude amplication.)

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 $\tilde{O}((\log N)s^2 t_0)$.

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 $\tilde{O}((\log N)s^2 t_0)$.
- Phase estimation produces error of $O(1/t_0)$ with tail probability dying off fast enough to not bother us.

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 $\tilde{O}((\log N)s^2 t_0)$.
- 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 $t_0 \sim \kappa^2/\epsilon$.

Aram Harrow

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 $\tilde{O}((\log N)s^2 t_0)$.
- 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.)

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 $\tilde{O}((\log N)s^2 t_0)$.
- 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 $O(\kappa^2/\epsilon)$.

Aram Harrow

University of

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 $\tilde{O}((\log N)s^2 t_0)$.
- 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 $O(\kappa^2/\epsilon)$.
- We couldn't figure out how to make variable-length run-time *à la* 0811.4428 work. Our best lower bound is $\sqrt{\kappa}$.

Algorithm details

Let $|b\rangle = \sum_\lambda b_\lambda |u_\lambda\rangle$.

1. Prepare control register in superposition of $|t\rangle$ over $0 \leq t \leq t_0$.
2. Use Hamiltonian simulation to apply $\sum_t |t\rangle\langle t| \otimes e^{iAt}$.
3. Fourier transform first register, yielding

$$\sum_{\lambda,\tilde{\lambda}} \alpha_{\lambda,\tilde{\lambda}} \left|\tilde{\lambda}\right\rangle \otimes b_\lambda |u_\lambda\rangle ,$$

   with $|\alpha_{\lambda,\tilde{\lambda}}|$ small unless $\tilde{\lambda} \approx \lambda$.
4. Conditioned on $\tilde{\lambda}$, adjoin state

$$\sqrt{1 - C^2\tilde{\lambda}^{-2}}\,|0\rangle + C\tilde{\lambda}^{-1}\,|1\rangle .$$

5. Undo steps 1-3
6. Measure ancilla qubit and start over if outcome isn't 1. (Technically, use amplitude amplication.)

Aram Harrow

University of

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 $\tilde{O}((\log N)s^2 t_0)$.
- Phase estimation produces error of $O(1/t_0)$ with tail probability dying off fast enough to not bother us.

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 $\tilde{O}((\log N)s^2 t_0)$.
- 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 $t_0 \sim \kappa^2/\epsilon$.

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 $\tilde{O}((\log N)s^2 t_0)$.
- 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 $O(\kappa^2/\epsilon)$.
- We couldn't figure out how to make variable-length run-time *à la* 0811.4428 work. Our best lower bound is $\sqrt{\kappa}$.

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 $\tilde{O}((\log N)s^2 t_0)$.
- 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 $O(\kappa^2/\epsilon)$.
- We couldn't figure out how to make variable-length run-time *à la* 0811.4428 work. Our best lower bound is $\sqrt{\kappa}$.

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.

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.

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.

Aram Harrow

University of

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 $\kappa = O(T^2)$.

Aram Harrow

University of

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 $\kappa = O(T^2)$.

Corollary

- A classical poly($\log(N), \kappa$) algorithm for estimating $\langle x| M |x\rangle$ to constant accuracy would imply BPP=BQP.
- Improving our quantum run-time to $\kappa^{\frac{1-\delta}{2}} \cdot \text{poly}\log(N)$ would imply that BQP=PSPACE.

Aram Harrow University of

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 $\kappa = O(T^2)$.

Corollary

- A classical poly($\log(N), \kappa$) algorithm for estimating $\langle x| M |x\rangle$ to constant accuracy would imply BPP=BQP.
- Improving our quantum run-time to $\kappa^{\frac{1-\delta}{2}} \cdot \text{poly}\log(N)$ would imply that BQP=PSPACE.

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 $-1 \le m_1, \ldots, m_N \le 1$, 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.

Aram Harrow

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 $\kappa = O(T^2)$.

Corollary

- A classical poly($\log(N), \kappa$) algorithm for estimating $\langle x| M |x\rangle$ to constant accuracy would imply BPP=BQP.
- Improving our quantum run-time to $\kappa^{\frac{1-\delta}{2}} \cdot \text{poly}\log(N)$ would imply that BQP=PSPACE.

Further consequences of BQP-completeness

Relative to oracles

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 $\kappa = O(T^2)$.

Corollary

- A classical poly$(\log(N), \kappa)$ algorithm for estimating $\langle x| M |x\rangle$ to constant accuracy would imply BPP=BQP.
- Improving our quantum run-time to $\kappa^{\frac{1-\delta}{2}} \cdot \mathrm{poly}\log(N)$ would imply that BQP=PSPACE.

Aram Harrow

University of

Further consequences of BQP-completeness

Relative to oracles

- No quantum algorithm can run in time $\kappa^{\frac{1-\delta}{2}} \cdot \mathrm{poly}\log(N)$.

Further consequences of BQP-completeness

Relative to oracles

- No quantum algorithm can run in time $\kappa^{\frac{1-\delta}{2}} \cdot \operatorname{poly}\log(N)$.
- No classical algorithm can run in time $N^{o(1)} 2^{o(\sqrt{\kappa})}$.

Further consequences of BQP-completeness

Relative to oracles

- No quantum algorithm can run in time $\kappa^{\frac{1-\delta}{2}} \cdot \operatorname{poly}\log(N)$.
- No classical algorithm can run in time $N^{o(1)} 2^{o(\sqrt{\kappa})}$.
- No iterative method can use $o(\sqrt{\kappa})$ matrix-vector multiplies. (Although we already knew this by taking A to be the adjacency matrix of a random cycle of length $\sqrt{\kappa}$.).

Aram Harrow

University of

Further consequences of BQP-completeness

Relative to oracles

- No quantum algorithm can run in time $\kappa^{\frac{1-\delta}{2}} \cdot \text{poly}\log(N)$.
- No classical algorithm can run in time $N^{o(1)}2^{o(\sqrt{\kappa})}$.
- No iterative method can use $o(\sqrt{\kappa})$ matrix-vector multiplies. (Although we already knew this by taking A to be the adjacency matrix of a random cycle of length $\sqrt{\kappa}$.).

Error scaling

- Improving our quantum run-time to $\text{poly}(\kappa, \log(N), \log(1/\epsilon))$ would imply BQP=PP.

Aram Harrow

University of

Further consequences of BQP-completeness

Relative to oracles

- No quantum algorithm can run in time $\kappa^{\frac{1-\delta}{2}} \cdot \mathrm{poly}\log(N)$.
- No classical algorithm can run in time $N^{o(1)} 2^{o(\sqrt{\kappa})}$.
- No iterative method can use $o(\sqrt{\kappa})$ matrix-vector multiplies. (Although we already knew this by taking A to be the adjacency matrix of a random cycle of length $\sqrt{\kappa}$.).

Error scaling

- Improving our quantum run-time to $\mathrm{poly}(\kappa, \log(N), \log(1/\epsilon))$ would imply BQP=PP.
- And even improving it to $N^{o(1)}/\epsilon^{o(1)}$ is impossible relative to an oracle.

Proof of BQP-hardness

An idea that almost works

- Our quantum circuit is $U_T \cdots U_1$.

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

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

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

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

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

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

- 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.

Aram Harrow

University of

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 $T \leq t < 2T$.

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

$$V = \sum_{t=1}^{3T} |t + 1 \pmod{3T}\rangle \langle t| \otimes U_t$$
$$A = I - e^{-\frac{1}{T}} V$$

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 $T \le t < 2T$.

- 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 $T \le t < 2T$) and there is no extra error from wrap-around.

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

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

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

- 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 $T \leq t < 2T$.

Aram Harrow

University of

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 polylogarithmic in the number of variables, but exponential in the integration time.
- [S. P. Jordan and P. Wocjan. Efficient quantum circuits for arbitrary sparse unitaries. arXiv: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. arXiv:quant-ph/0606229] is similar to our BQP-hardness result.

Extensions/applications

Mostly things we don't know how to solve!

- 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 $\ll 1/\kappa$.
  However, we cannot determine exactly which eigenvalues are $> 1/\kappa$ and which are $< 1/\kappa$.

Extensions/applications

Mostly things we don't know how to solve!

- 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 $\ll 1/\kappa$.
  However, we cannot determine exactly which eigenvalues are $> 1/\kappa$ and which are $< 1/\kappa$.
- 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 appears to require more careful analysis of errors in Hamiltonian simulation protocols.

Extensions/applications

Mostly things we don't know how to solve!

- 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 $\ll 1/\kappa$.
  However, we cannot determine exactly which eigenvalues are $> 1/\kappa$ and which are $< 1/\kappa$.
- 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 appears to require more careful analysis of errors in Hamiltonian simulation protocols.
- 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.

Extensions/applications

Mostly things we don't know how to solve!

- 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 $\ll 1/\kappa$.
  However, we cannot determine exactly which eigenvalues are $> 1/\kappa$ and which are $< 1/\kappa$.
- 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 appears to require more careful analysis of errors in Hamiltonian simulation protocols.
- 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.
- Future work. Find applications! Candidates are deconvolution, solving elliptical PDE's and speeding up linear programming.