Quantum algorithms: What's quantum complexity theory got to do with it?

Sevag Gharibian

Department of Computer Science
Institute for Photonic Quantum Systems (PhoQS)
Paderborn University
Germany

Mindset

Quantum computers

Mindset

Quantum computers

vs.

Classical computers

Mindset

Quantum computers

vs.

Classical computers

Mindset

Quantum computers

vs.

Classical computers

Comparison:

1. Classical: Honed over decades, extremely good at many tasks. Actually exist.

Mindset

Quantum computers

vs.

Classical computers

Comparison:

1. Classical: Honed over decades, extremely good at many tasks. Actually exist.
2. Quantum:
   - Promise* of substantial improved performance for certain (important!) tasks.

Mindset

Quantum computers

vs.

Classical computers

Comparison:

1. Classical: Honed over decades, extremely good at many tasks. Actually exist.
2. Quantum:
   - Promise* of substantial improved performance for certain (important!) tasks.
   - Relatively early in hardware development

With an eye on...

Computational Complexity Theory

- What resources (e.g. time, space) required to solve a computational problem?
- Complexity classes such as P, NP, BQP, QMA

Outline

Origins

Wiesner's quantum money (late 1970's): "Unforgeable" notion of money

- Security: Proven via semidefinite programming [MVW12]
- Adaptive attack: Scheme insecure if bank *returns* banknote after checking it! [BNSU14]

Origins

Wiesner's quantum money (late 1970's): "Unforgeable" notion of money

- Security: Proven via semidefinite programming [MVW12]
- Adaptive attack: Scheme insecure if bank *returns* banknote after checking it! [BNSU14]

Encapsulates many traits of quantum computing:

- Possible to do things which are impossible classically (due, e.g., to no-cloning theorem)
- Such feats do *not* require a universal quantum computer
- Beware the details hiding in quantum claims

The 1990's: The first "gamechangers"

Shor (1994)

- Poly-time quantum algorithm for integer factorization
- Breaks popular cryptosystem RSA, whose security assumes "classical hardness" of factoring

The 1990's: The first "gamechangers"

Shor (1994)

- Poly-time quantum algorithm for integer factorization
- Breaks popular cryptosystem RSA, whose security assumes "classical hardness" of factoring
- Complexity of factoring: Believed "NP-intermediate", i.e. neither in P nor NP-complete

The 1990's: The first "gamechangers"

Shor (1994)

- Poly-time quantum algorithm for integer factorization
- Breaks popular cryptosystem RSA, whose security assumes "classical hardness" of factoring
- Complexity of factoring: Believed "NP-intermediate", i.e. neither in P nor NP-complete
- Other famous NP-intermediate problem: Graph Isomorphism (GI)
  - Shor's period-finding generalized to "hidden subgroup" problem with hope of solving GI

The 1990's: The first "gamechangers"

Shor (1994)

- Poly-time quantum algorithm for integer factorization
- Breaks popular cryptosystem RSA, whose security assumes "classical hardness" of factoring
- Complexity of factoring: Believed "NP-intermediate", i.e. neither in P nor NP-complete
- Other famous NP-intermediate problem: Graph Isomorphism (GI)
  - Shor's period-finding generalized to "hidden subgroup" problem with hope of solving GI
  - Surprise: GI has a *quasi*-poly-time classical algorithm [Babai 2016]

The 1990's: The first "gamechangers"

Shor (1994)

- Poly-time quantum algorithm for integer factorization
- Breaks popular cryptosystem RSA, whose security assumes "classical hardness" of factoring
- Complexity of factoring: Believed "NP-intermediate", i.e. neither in P nor NP-complete
- Other famous NP-intermediate problem: Graph Isomorphism (GI)
  - Shor's period-finding generalized to "hidden subgroup" problem with hope of solving GI
  - Surprise: GI has a *quasi*-poly-time classical algorithm [Babai 2016]

The 1990's: The first "gamechangers"

Grover (1996)

- Finds marked item in unstructured database of N items with $O(\sqrt{N})$ queries
- Generalized to amplitude amplification:
  - Boosts any probabilistic algorithm with success probability p to success probability $\sqrt{p}$

The 1990's: The first "gamechangers"

Grover (1996)

- Finds marked item in unstructured database of N items with $O(\sqrt{N})$ queries
- Generalized to amplitude amplification:
  - Boosts any probabilistic algorithm with success probability p to success probability $\sqrt{p}$

Lloyd's Hamiltonian simulation algorithm (1996)

- Efficiently simulates quantum systems governed by local Hamiltonians $H = \sum_i H_i \in \mathcal{L}(\mathbb{C}^2)^{\otimes n}$
- Introduced use of Trotterization/Lie Product Formula:

$$e^{H_1+H_2} = \lim_{n\to\infty} \left(e^{\frac{H_1}{n}} e^{\frac{H_2}{n}} \right)^n .$$

The 2000's: The posterchild

Harrow-Hassadim-Lloyd (HHL) algorithm (2008)

- Solves* linear systems of equations $Ax = b$ with exponential speedup, i.e. time polylog$(\dim(x))$

The 2000's: The posterchild

Harrow-Hassadim-Lloyd (HHL) algorithm (2008)

- Solves* linear systems of equations $Ax = b$ with exponential speedup, i.e. time $\text{polylog}(\dim(x))$

- *: Returns quantum representation $|\psi_x\rangle \in (\mathbb{C}^2)^{\otimes n}$ of x $\Rightarrow$ can't read all entries of x!

The 2000's: The posterchild

Harrow-Hassadim-Lloyd (HHL) algorithm (2008)

- Solves* linear systems of equations $Ax = b$ with exponential speedup, i.e. time polylog$(\mathrm{dim}(x))$

- *: Returns quantum representation $|\psi_x\rangle \in (\mathbb{C}^2)^{\otimes n}$ of $x \Rightarrow$ can't read all entries of x!
- Approach: "Eigenvalue surgery"
  1. Use Hamiltonian simulation to simulate unitary $U = e^{iA}$
  2. Use Quantum Phase Estimation on U to "extract" eigenvalues of A and "manually" invert them

The 2000's: The posterchild

Harrow-Hassadim-Lloyd (HHL) algorithm (2008)

- Solves* linear systems of equations $Ax = b$ with exponential speedup, i.e. time polylog$(\dim(x))$

- *: Returns quantum representation $|\psi_x\rangle \in (\mathbb{C}^2)^{\otimes n}$ of $x \Rightarrow$ can't read all entries of x!
- Approach: "Eigenvalue surgery"
  1. Use Hamiltonian simulation to simulate unitary $U = e^{iA}$
  2. Use Quantum Phase Estimation on U to "extract" eigenvalues of A and "manually" invert them
- BQP-complete: Matrix inversion precisely captures the power of efficient quantum computation

The 2010's: A general quantum algorithms framework

Low-Chuang optimal Hamiltonian simulation algorithm (2016)

- Simulate Hamiltonian H for time t and error ϵ, i.e. unitary $U = e^{iHt}$ in time $O(t + \log(1/\epsilon))$

The 2010's: A general quantum algorithms framework

Low-Chuang optimal Hamiltonian simulation algorithm (2016)

- Simulate Hamiltonian H for time t and error ϵ, i.e. unitary $U = e^{iHt}$ in time $O(t + \log(1/\epsilon))$
- Introduced technique of qubitization or "block encodings":

$$U = \begin{pmatrix} H & M_{12} & \cdots & M_{1m} \\ M_{21} & M_{22} & \cdots & M_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mm} \end{pmatrix} \rightarrow U' = \begin{pmatrix} p(H) & M_{12} & \cdots & M_{1m} \\ M_{21} & M_{22} & \cdots & M_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mm} \end{pmatrix}$$

- Idea:
  - Want to map $|\psi\rangle \mapsto e^{iHt}|\psi\rangle$

The 2010's: A general quantum algorithms framework

Low-Chuang optimal Hamiltonian simulation algorithm (2016)

- Simulate Hamiltonian H for time t and error ϵ, i.e. unitary $U = e^{iHt}$ in time $O(t + \log(1/\epsilon))$
- Introduced technique of qubitization or "block encodings":

$$U = \begin{pmatrix} H & M_{12} & \cdots & M_{1m} \\ M_{21} & M_{22} & \cdots & M_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mm} \end{pmatrix} \rightarrow U' = \begin{pmatrix} p(H) & M_{12} & \cdots & M_{1m} \\ M_{21} & M_{22} & \cdots & M_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mm} \end{pmatrix}$$

- Idea:
  - Want to map $|\psi\rangle \mapsto e^{iHt}|\psi\rangle$
  - Embed H in top-left block of some unitary U

The 2010's: A general quantum algorithms framework

Low-Chuang optimal Hamiltonian simulation algorithm (2016)

- Simulate Hamiltonian H for time t and error ϵ, i.e. unitary $U = e^{iHt}$ in time $O(t + \log(1/\epsilon))$
- Introduced technique of qubitization or "block encodings":

$$U = \begin{pmatrix} H & M_{12} & \cdots & M_{1m} \\ M_{21} & M_{22} & \cdots & M_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mm} \end{pmatrix} \quad \rightarrow \quad U' = \begin{pmatrix} p(H) & M_{12} & \cdots & M_{1m} \\ M_{21} & M_{22} & \cdots & M_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mm} \end{pmatrix}$$

- Idea:
  - Want to map $|\psi\rangle \mapsto e^{iHt}|\psi\rangle$
  - Embed H in top-left block of some unitary U
  - Use "qubitization" to map $H \mapsto p(H)$ for some appropriate polynomial p such that $p(H) \approx e^{iHt}$

The 2010's: A general quantum algorithms framework

Low-Chuang optimal Hamiltonian simulation algorithm (2016)

- Simulate Hamiltonian H for time t and error ϵ, i.e. unitary $U = e^{iHt}$ in time $O(t + \log(1/\epsilon))$
- Introduced technique of qubitization or "block encodings":

$$U = \begin{pmatrix} H & M_{12} & \cdots & M_{1m} \\ M_{21} & M_{22} & \cdots & M_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mm} \end{pmatrix} \rightarrow U' = \begin{pmatrix} p(H) & M_{12} & \cdots & M_{1m} \\ M_{21} & M_{22} & \cdots & M_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mm} \end{pmatrix}$$

- Idea:
  - Want to map $|\psi\rangle \mapsto e^{iHt}|\psi\rangle$
  - Embed H in top-left block of some unitary U
  - Use "qubitization" to map $H \mapsto p(H)$ for some appropriate polynomial p such that $p(H) \approx e^{iHt}$
  - Use "post-selection" to probabilistically map

$$U|\psi\rangle \mapsto p(H)|\psi\rangle \approx e^{iHt}|\psi\rangle.$$

The 2010's: A general quantum algorithms framework

Quantum Singular Value Transformation (Gilyén, Su, Low, and Wiebe 2019)

- Generalizes Low and Chuang's qubitization approach to *non-square* matrices A
- Given non-square A (embedded as block of unitary U), polynomial p, simulates mapping

$$|\psi\rangle \mapsto p\left(\sqrt{A^\dagger A}\right)|\psi\rangle.$$

The 2010's: A general quantum algorithms framework

Quantum Singular Value Transformation (Gilyén, Su, Low, and Wiebe 2019)

- Generalizes Low and Chuang's qubitization approach to *non-square* matrices A
- Given non-square A (embedded as block of unitary U), polynomial p, simulates mapping

$$|\psi\rangle \mapsto p\left(\sqrt{A^\dagger A}\right)|\psi\rangle.$$

- Unified framework for a host of quantum algorithms:
  - Hamiltonian simulation, linear systems, amplitude amplification, quantum machine learning algorithms, and essentially all "quantum matrix linear algebra"

The 2010's: A general quantum algorithms framework

Quantum Singular Value Transformation (Gilyén, Su, Low, and Wiebe 2019)

- Generalizes Low and Chuang's qubitization approach to *non-square* matrices A
- Given non-square A (embedded as block of unitary U), polynomial p, simulates mapping

$$|\psi\rangle \mapsto p\left(\sqrt{A^\dagger A}\right)|\psi\rangle.$$

- Unified framework for a host of quantum algorithms:
  - Hamiltonian simulation, linear systems, amplitude amplification, quantum machine learning algorithms, and essentially all "quantum matrix linear algebra"
  - [Martyn, Rossi, Tan and Chuang 2021] *iteratively* apply QSVT to simulate Fourier Transform

Outline

Which computing model?

Universal models:

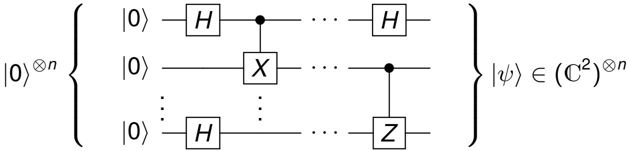
- Quantum Turing Machines
- Quantum circuits
- Quantum adiabatic computing
- One-way measurement based computing
- Quantum walks
- Quantum Approximate Optimization Algorithm (QAOA)

Which computing model?

Universal models:

- Quantum Turing Machines
- Quantum circuits
- Quantum adiabatic computing
- One-way measurement based computing
- Quantum walks
- Quantum Approximate Optimization Algorithm (QAOA)

Here: Work with $\mathrm{poly}(n)$-size quantum circuit implementing n-qubit unitaries U, e.g.

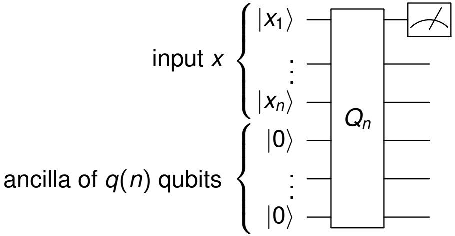


What counts as an efficient quantum algorithm?

Bounded-error quantum polynomial-time (BQP)

Promise problem $\mathbb{A} = (A_{\mathrm{yes}}, A_{\mathrm{no}}) \in \mathrm{BQP}$ if $\exists$ P-uniform quantum circuit family $\{Q_n\}$ and polynomial q as below. The first output qubit of Q_n is measured in the standard basis and returned. For any input $x \in \{0,1\}^*$:

- (YES case) If $x \in A_{\mathrm{yes}}$, then Q_n outputs 1 with probability at least $2/3$.
- (NO case) If $x \in A_{\mathrm{no}}$, then Q_n outputs 1 with probability at most $1/3$.

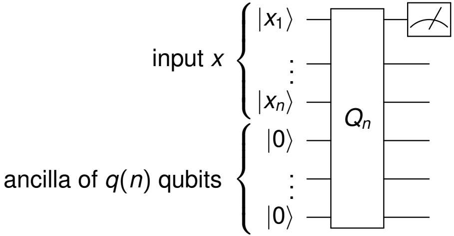


What counts as an efficient quantum algorithm?

Bounded-error quantum polynomial-time (BQP)

Promise problem $\mathbb{A} = (A_{\text{yes}}, A_{\text{no}}) \in \text{BQP}$ if $\exists$ P-uniform quantum circuit family $\{Q_n\}$ and polynomial q as below. The first output qubit of Q_n is measured in the standard basis and returned. For any input $x \in \{0,1\}^*$:

- (YES case) If $x \in A_{\text{yes}}$, then Q_n outputs 1 with probability at least $2/3$.
- (NO case) If $x \in A_{\text{no}}$, then Q_n outputs 1 with probability at most $1/3$.



Exercise: Why do we require a P-uniform quantum circuit family?

Assumption

All quantum operations are noise-free, i.e. perfect

Outline

The Pikachu of BQP

Linear system solving:

- Input: Invertible $A \in \mathbb{C}^{N\times N}$ and target vector $\mathbf{b} \in \mathbb{C}^N$
- Output: $\mathbf{x} \in \mathbb{C}^N$ such that $A\mathbf{x} = \mathbf{b}$.

What is the complexity of linear system solving?

The Pikachu of BQP

Linear system solving:

- Input: Invertible $A \in \mathbb{C}^{N\times N}$ and target vector $\mathbf{b} \in \mathbb{C}^N$
- Output: $\mathbf{x} \in \mathbb{C}^N$ such that $A\mathbf{x} = \mathbf{b}$.

What is the complexity of linear system solving?

- If A and $\mathbf{x}$ given explicitly in matrix form $\Rightarrow$ $\mathbf{x} = A^{-1}\mathbf{b}$ classically in time $\mathrm{poly}(N)$

The Pikachu of BQP

Linear system solving:

- Input: Invertible $A \in \mathbb{C}^{N\times N}$ and target vector $\mathbf{b} \in \mathbb{C}^N$
- Output: $\mathbf{x} \in \mathbb{C}^N$ such that $A\mathbf{x} = \mathbf{b}$.

What is the complexity of linear system solving?

- If A and $\mathbf{x}$ given explicitly in matrix form $\Rightarrow \mathbf{x} = A^{-1}\mathbf{b}$ classically in time $\mathrm{poly}(N)$
- If A represented "succinctly" via query-access and $\mathbf{b}$ given via quantum circuit?

Matrix inversion problem (MI)

Input: $O(1)$-sparse row-computable invertible Hermitian matrix $A \in \mathbb{C}^{N\times N}$.

- $O(1)$-sparse: At most $O(1)$ non-zero entries per row.
- Row-computable: $\exists$ polylog(N)-time classical algorithm which, given $r \in [N]$, outputs entries of row r.

Matrix inversion problem (MI)

Input: $O(1)$-sparse row-computable invertible Hermitian matrix $A \in \mathbb{C}^{N\times N}$.

- $O(1)$-sparse: At most $O(1)$ non-zero entries per row.
- Row-computable: $\exists$ polylog(N)-time classical algorithm which, given $r \in [N]$, outputs entries of row r.

Output: Let $|x\rangle \propto A^{-1}|0^N\rangle$ be a unit vector, and $\Pi = |1\rangle\langle 1|$ a projector onto the first qubit of $|x\rangle$. Then:

Matrix inversion problem (MI)

Input: $O(1)$-sparse row-computable invertible Hermitian matrix $A \in \mathbb{C}^{N \times N}$.

- $O(1)$-sparse: At most $O(1)$ non-zero entries per row.
- Row-computable: $\exists$ polylog(N)-time classical algorithm which, given $r \in [N]$, outputs entries of row r.

Output: Let $|x\rangle \propto A^{-1}|0^N\rangle$ be a unit vector, and $\Pi = |1\rangle\langle 1|$ a projector onto the first qubit of $|x\rangle$. Then:

- If $\langle x|\Pi|x\rangle \geq 2/3$, output YES.
- If $\langle x|\Pi|x\rangle \leq 1/3$, output NO.

Matrix inversion problem (MI)

Input: $O(1)$-sparse row-computable invertible Hermitian matrix $A \in \mathbb{C}^{N\times N}$.

- $O(1)$-sparse: At most $O(1)$ non-zero entries per row.
- Row-computable: $\exists$ polylog(N)-time classical algorithm which, given $r \in [N]$, outputs entries of row r.

Output: Let $|x\rangle \propto A^{-1}|0^N\rangle$ be a unit vector, and $\Pi = |1\rangle\langle 1|$ a projector onto the first qubit of $|x\rangle$. Then:

- If $\langle x|\Pi|x\rangle \geq 2/3$, output YES.
- If $\langle x|\Pi|x\rangle \leq 1/3$, output NO.

Theorem [Harrow, Hassidim, Lloyd, 2008]

MI is BQP-complete under poly-time many-one reduction, i.e.:

Matrix inversion problem (MI)

Input: $O(1)$-sparse row-computable invertible Hermitian matrix $A \in \mathbb{C}^{N\times N}$.

- $O(1)$-sparse: At most $O(1)$ non-zero entries per row.
- Row-computable: $\exists$ polylog(N)-time classical algorithm which, given $r \in [N]$, outputs entries of row r.

Output: Let $|x\rangle \propto A^{-1}|0^N\rangle$ be a unit vector, and $\Pi = |1\rangle\langle 1|$ a projector onto the first qubit of $|x\rangle$. Then:

- If $\langle x|\Pi|x\rangle \geq 2/3$, output YES.
- If $\langle x|\Pi|x\rangle \leq 1/3$, output NO.

Theorem [Harrow, Hassidim, Lloyd, 2008]

MI is BQP-complete under poly-time many-one reduction, i.e.:

- MI is in BQP, i.e. can be efficiently solved in polylog(N) time on a quantum computer,

Matrix inversion problem (MI)

Input: $O(1)$-sparse row-computable invertible Hermitian matrix $A \in \mathbb{C}^{N \times N}$.

- $O(1)$-sparse: At most $O(1)$ non-zero entries per row.
- Row-computable: $\exists$ polylog(N)-time classical algorithm which, given $r \in [N]$, outputs entries of row r.

Output: Let $|x\rangle \propto A^{-1}|0^N\rangle$ be a unit vector, and $\Pi = |1\rangle\langle 1|$ a projector onto the first qubit of $|x\rangle$. Then:

- If $\langle x|\Pi|x\rangle \geq 2/3$, output YES.
- If $\langle x|\Pi|x\rangle \leq 1/3$, output NO.

Theorem [Harrow, Hassidim, Lloyd, 2008]

MI is BQP-complete under poly-time many-one reduction, i.e.:

- MI is in BQP, i.e. can be efficiently solved in polylog(N) time on a quantum computer,
- MI is BQP-hard, i.e. BQP computation can be reduced to an instance of MI.

Outline

Overview

Goal: Given sparse Hermitian A and poly-size circuit for $|b\rangle$, want to compute unit vector $|x\rangle \propto A^{-1}|b\rangle$.

Overview

Goal: Given sparse Hermitian A and poly-size circuit for $|b\rangle$, want to compute unit vector $|x\rangle \propto A^{-1}|b\rangle$.

Idea: To compute A^{-1}, *coherently invert* each eigenvalue of A via Quantum Phase Estimation (QPE).

Notation: Spectral decomposition $A = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|$.

Overview

Goal: Given sparse Hermitian A and poly-size circuit for $|b\rangle$, want to compute unit vector $|x\rangle \propto A^{-1}|b\rangle$.

Idea: To compute A^{-1}, *coherently invert* each eigenvalue of A via Quantum Phase Estimation (QPE).

Notation: Spectral decomposition $A = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|$.

Framework: Eigenvalue surgery

1. Eigenvalue extraction (via Hamiltonian simulation and Quantum Phase Estimation (QPE))
2. Eigenvalue processing (done classically, coherently)
3. Eigenvalue reinsertion (via postselection)

Hamiltonian simulation

Question: Why is quantum dynamics *unitary*?

Hamiltonian simulation

Question: Why is quantum dynamics *unitary*?

(Time-independent) Schrödinger equation

Time evolution of any n-qubit system governed by Hermitian matrix $H \in \mathcal{L}(\mathbb{C}^2)^{\otimes n}$, called a Hamiltonian:

$$i\frac{d|\psi\rangle}{dt} = H|\psi\rangle$$

Hamiltonian simulation

Question: Why is quantum dynamics *unitary*?

(Time-independent) Schrödinger equation

Time evolution of any n-qubit system governed by Hermitian matrix $H \in \mathcal{L}(\mathbb{C}^2)^{\otimes n}$, called a Hamiltonian:

$$i\frac{d|\psi\rangle}{dt} = H|\psi\rangle \quad \xrightarrow{\text{solve}} \quad |\psi_t\rangle = e^{-iHt}|\psi_0\rangle \quad (\leftarrow \text{ unitary!})$$

Hamiltonian simulation

Question: Why is quantum dynamics *unitary*?

(Time-independent) Schrödinger equation

Time evolution of any n-qubit system governed by Hermitian matrix $H \in \mathcal{L}(\mathbb{C}^2)^{\otimes n}$, called a Hamiltonian:

$$i\frac{d|\psi\rangle}{dt} = H|\psi\rangle \quad \xrightarrow{\text{solve}} \quad |\psi_t\rangle = e^{-iHt}|\psi_0\rangle \quad (\leftarrow \text{ unitary!})$$

Hamiltonian simulation [Low, Chuang 2017]

Given d-sparse H, simulation time $t \geq 0$, and $\epsilon > 0$, can simulate e^{iHt} up to error ϵ and success probability at least $1 - 2\epsilon$ in time[a]

$$O\left(td\,\|H\|_{\max} + \frac{\log(1/\epsilon)}{\log\log(1/\epsilon)}\right).$$

[a]Query complexity. Gate complexity has $O(n)$ overhead.

Overview

Goal: Given sparse Hermitian A and poly-size circuit for $|b\rangle$, want to compute unit vector $|x\rangle \propto A^{-1}|b\rangle$.

Idea: To compute A^{-1}, *coherently invert* each eigenvalue of A via Quantum Phase Estimation (QPE).

Notation: Spectral decomposition $A = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|$.

Framework: Eigenvalue surgery

1. Eigenvalue extraction (via Hamiltonian simulation and Quantum Phase Estimation (QPE))
2. Eigenvalue processing (done classically, coherently)
3. Eigenvalue reinsertion (via postselection)

Quantum Phase Estimation (QPE)

- Hermitian H with spectral decomposition $H = \sum_j \lambda_j |\psi_j\rangle\langle\psi_j|$ acting on n qubits.
- Spectral decomposition of corresponding Hamiltonian evolution/unitary:

$$U = e^{iH} = \sum_j e^{i\lambda_j} |\psi_j\rangle\langle\psi_j|.$$

Quantum Phase Estimation (QPE)

- Hermitian H with spectral decomposition $H = \sum_j \lambda_j |\psi_j\rangle\langle\psi_j|$ acting on n qubits.
- Spectral decomposition of corresponding Hamiltonian evolution/unitary:

$$U = e^{iH} = \sum_j e^{i\lambda_j} |\psi_j\rangle\langle\psi_j|.$$

- Goal: Given eigenvector $|\psi_j\rangle$, precision parameter k, want to compute λ_j to k bits of precision.

Quantum Phase Estimation (QPE)

- Hermitian H with spectral decomposition $H = \sum_j \lambda_j |\psi_j\rangle\langle\psi_j|$ acting on n qubits.
- Spectral decomposition of corresponding Hamiltonian evolution/unitary:

$$U = e^{iH} = \sum_j e^{i\lambda_j} |\psi_j\rangle\langle\psi_j|.$$

- Goal: Given eigenvector $|\psi_j\rangle$, precision parameter k, want to compute λ_j to k bits of precision.

Quantum Phase Estimation (QPE)

Given precision k, and ability to compute controlled-U^{2^K} for $1 \leq K \leq k$ in time $\mathrm{poly}(n)$, map

$$|0^k\rangle|\psi_j\rangle \mapsto |\widetilde{\lambda}_j\rangle|\psi_j\rangle$$

in time $\mathrm{poly}(n)$, where $\widetilde{\lambda}_j$ is λ_j up to k bits.

Quantum Phase Estimation (QPE)

- Hermitian H with spectral decomposition $H = \sum_j \lambda_j |\psi_j\rangle\langle\psi_j|$ acting on n qubits.
- Spectral decomposition of corresponding Hamiltonian evolution/unitary:

$$U = e^{iH} = \sum_j e^{i\lambda_j} |\psi_j\rangle\langle\psi_j|.$$

- Goal: Given eigenvector $|\psi_j\rangle$, precision parameter k, want to compute λ_j to k bits of precision.

Quantum Phase Estimation (QPE)

Given precision k, and ability to compute controlled-U^{2^K} for $1 \leq K \leq k$ in time $\text{poly}(n)$, map

$$|0^k\rangle|\psi_j\rangle \mapsto |\widetilde{\lambda}_j\rangle|\psi_j\rangle$$

in time $\text{poly}(n)$, where $\widetilde{\lambda}_j$ is λ_j up to k bits.

Exercise: Given n-qubit unitary U, can we efficiently compute U^{2^n} in general?

Overview

Goal: Given sparse Hermitian A and poly-size circuit for $|b\rangle$, want to compute unit vector $|x\rangle \propto A^{-1}|b\rangle$.

Idea: To compute A^{-1}, *coherently invert* each eigenvalue of A via Quantum Phase Estimation (QPE).

Notation: Spectral decomposition $A = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|$.

Framework: Eigenvalue surgery

1. Eigenvalue extraction (via Hamiltonian simulation and Quantum Phase Estimation (QPE))
2. Eigenvalue processing (done classically, coherently)
3. Eigenvalue reinsertion (via postselection)

Step 1: Eigenvalue extraction

Recall spectral decomposition $A = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|$.

- Prepare target state

$$|b\rangle = \sum_{j=1}^{N} \alpha_j |\psi_j\rangle \in \mathbb{C}^N,$$

for eigenvectors $|\psi_j\rangle$ of A. (Recall: Given circuit to prepare $|b\rangle$ as input.)

Step 1: Eigenvalue extraction

Recall spectral decomposition $A = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|$.

- Prepare target state
$$|b\rangle = \sum_{j=1}^{N} \alpha_j |\psi_j\rangle \in \mathbb{C}^N,$$
for eigenvectors $|\psi_j\rangle$ of A. (Recall: Given circuit to prepare $|b\rangle$ as input.)

- Apply QPE to unitary e^{iA} with an n-qubit ancilla:
$$\sum_{j=1}^{N} \alpha_j |0^n\rangle |\psi_j\rangle \quad \mapsto \quad \sum_{j=1}^{N} \alpha_j |\lambda_j\rangle |\psi_j\rangle \in (\mathbb{C}^2)^{\otimes n} \otimes \mathbb{C}^N.$$

Step 2: Eigenvalue processing

- Conditioned on the first register, rotate a new single-qubit ancilla as follows:

$$\sum_{j=1}^{N}\alpha_j|\lambda_j\rangle|\psi_j\rangle|0\rangle \quad\mapsto\quad \sum_{j=1}^{N}\alpha_j|\lambda_j\rangle|\psi_j\rangle\left(\sqrt{1-\frac{1}{\lambda_j^2\kappa^2(A)}}|0\rangle+\left(\frac{1}{\lambda_j\kappa(A)}\right)|1\rangle\right)\in(\mathbb{C}^2)^{\otimes n}\otimes\mathbb{C}^N\otimes\mathbb{C}^2.$$

Key parameter: Condition number $\kappa(A) := \|A^-1\|_\infty\|A\|_\infty$.

Step 2: Eigenvalue processing

- Conditioned on the first register, rotate a new single-qubit ancilla as follows:

$$\sum_{j=1}^{N} \alpha_j |\lambda_j\rangle|\psi_j\rangle|0\rangle \quad \mapsto \quad \sum_{j=1}^{N} \alpha_j |\lambda_j\rangle|\psi_j\rangle \left(\sqrt{1 - \frac{1}{\lambda_j^2 \kappa^2(A)}}|0\rangle + \left(\frac{1}{\lambda_j \kappa(A)}\right)|1\rangle \right) \in (\mathbb{C}^2)^{\otimes n} \otimes \mathbb{C}^N \otimes \mathbb{C}^2.$$

Key parameter: Condition number $\kappa(A) := \|A^-1\|_\infty \|A\|_\infty$.

Exercise:
Assume $\|A\|_\infty = 1$. Show

$$\frac{1}{\kappa(A)} \leq \frac{1}{\lambda_j \kappa(A)} \leq 1.$$

Thus, amplitudes above well-defined.

Step 3: Eigenvalue reinsertion

- Uncompute eigenvalues via inverse QPE:

$$\begin{aligned} &\sum_{j=1}^{N}\alpha_j{\color{red}|\lambda_j\rangle}|\psi_j\rangle\left(\sqrt{1-\frac{1}{\lambda_j^2\kappa^2(A)}}|0\rangle+\left(\frac{1}{\lambda_j\kappa(A)}\right)|1\rangle\right)\\ \mapsto\quad &\sum_{j=1}^{N}\alpha_j{\color{red}|0\rangle}|\psi_j\rangle\left(\sqrt{1-\frac{1}{\lambda_j^2\kappa^2(A)}}|0\rangle+\left(\frac{1}{\lambda_j\kappa(A)}\right)|1\rangle\right). \end{aligned}$$

Step 3: Eigenvalue reinsertion

- Uncompute eigenvalues via inverse QPE:

$$\sum_{j=1}^{N} \alpha_j |\lambda_j\rangle|\psi_j\rangle\left(\sqrt{1-\frac{1}{\lambda_j^2\kappa^2(A)}}|0\rangle+\left(\frac{1}{\lambda_j\kappa(A)}\right)|1\rangle\right)$$
$$\mapsto \quad \sum_{j=1}^{N} \alpha_j |0\rangle|\psi_j\rangle\left(\sqrt{1-\frac{1}{\lambda_j^2\kappa^2(A)}}|0\rangle+\left(\frac{1}{\lambda_j\kappa(A)}\right)|1\rangle\right).$$

- Measure third register in standard basis, postselect on outcome 1, discard third register:

$$\sum_{j=1}^{N} \alpha_j \left(\frac{1}{\lambda_j}\right)|\psi_j\rangle \propto A^{-1}|b\rangle \in \mathbb{C}^N.$$

Step 3: Eigenvalue reinsertion

- Uncompute eigenvalues via inverse QPE:

$$
\begin{aligned}
&\sum_{j=1}^{N}\alpha_j|\lambda_j\rangle|\psi_j\rangle\left(\sqrt{1-\frac{1}{\lambda_j^2\kappa^2(A)}}|0\rangle+\left(\frac{1}{\lambda_j\kappa(A)}\right)|1\rangle\right)\\
\mapsto\quad&\sum_{j=1}^{N}\alpha_j|0\rangle|\psi_j\rangle\left(\sqrt{1-\frac{1}{\lambda_j^2\kappa^2(A)}}|0\rangle+\left(\frac{1}{\lambda_j\kappa(A)}\right)|1\rangle\right).
\end{aligned}
$$

- Measure third register in standard basis, postselect on outcome 1, discard third register:

$$\sum_{j=1}^{N}\alpha_j\left(\frac{1}{\lambda_j}\right)|\psi_j\rangle\propto A^{-1}|b\rangle\in\mathbb{C}^N.$$

Exercise. Prove that probability of obtaining outcome 1 is at least $1/\kappa^2(A)$.

Runtime

To compute unit vector proportional to $A^{-1}|b\rangle$ within error ϵ:

$$\widetilde{O}(\log(N)s^2\kappa^2(A)/\epsilon) \quad \text{where}$$

- N the dimension of A,
- s the sparsity of A,
- $\log N$ the number of qubits A acts on.

Runtime

To compute unit vector proportional to $A^{-1}|b\rangle$ within error ϵ:

$$\widetilde{O}(\log(N)s^2\kappa^2(A)/\epsilon) \quad \text{where}$$

- N the dimension of A,
- s the sparsity of A,
- $\log N$ the number of qubits A acts on.

Implication:

- When $\kappa(A), s \in \text{polylog}(N)$, exponentially faster than classically solving $N \times N$ system.
- But this solves a different problem than classical linear systems solvers!

Outline

Matrix inversion problem (MI)

Input: $O(1)$-sparse row-computable invertible Hermitian matrix $A \in \mathbb{C}^{N\times N}$

- $O(1)$-sparse: At most $O(1)$ non-zero entries per row.
- Row-computable: $\exists$ polylog(N)-time classical algorithm which, given $r \in [N]$, outputs entries of row r.

Output: Let $|x\rangle \propto A^{-1}|0^N\rangle$ be a unit vector, and $\Pi = |1\rangle\langle 1|$ a projector onto the first qubit of $|x\rangle$. Then:

- If $\langle x|\Pi|x\rangle \geq 2/3$, output YES.
- If $\langle x|\Pi|x\rangle \leq 1/3$, output NO.

Theorem [Harrow, Hassidim, Lloyd, 2008]

MI is BQP-complete under poly-time many-one reduction, i.e.:

- MI is in BQP, i.e. can be efficiently solved in polylog(N) time on a quantum computer,
- MI is BQP-hard, i.e. BQP computation can be reduced to an instance of MI.

MI is BQP-hard

Goal: Show that any BQP computation V poly-time reducible to an instance A of MI.

Arbitrary BQP circuit

MI is BQP-hard

Goal: Show that any BQP computation V poly-time reducible to an instance A of MI.

Arbitrary BQP circuit

MI instance

MI is BQP-hard

Goal: Show that any BQP computation V poly-time reducible to an instance A of MI.

Arbitrary BQP circuit

MI instance

Moral: If you can solve MI, you can simulate any BQP circuit

MI is BQP-hard

Goal: Show that any BQP computation V poly-time reducible to an instance A of MI.

Starting point: Let $V = V_m \cdots V_1$ be a BQP circuit on n qubits, $N = 2^n$. Assume WLOG m is power of 2.

Problem: Need to tie matrix inverse with action of V.

MI is BQP-hard

Goal: Show that any BQP computation V poly-time reducible to an instance A of MI.

Starting point: Let $V = V_m \cdots V_1$ be a BQP circuit on n qubits, $N = 2^n$. Assume WLOG m is power of 2.

Problem: Need to tie matrix inverse with action of V.

Idea:

- Recall Maclaurin series $\frac{1}{1-x} = \sum_{l=0}^{\infty} x^l$ for $|x| < 1$.

MI is BQP-hard

Goal: Show that any BQP computation V poly-time reducible to an instance A of MI.

Starting point: Let $V = V_m \cdots V_1$ be a BQP circuit on n qubits, $N = 2^n$. Assume WLOG m is power of 2.

Problem: Need to tie matrix inverse with action of V.

Idea:

- Recall Maclaurin series $\frac{1}{1-x} = \sum_{l=0}^{\infty} x^l$ for $|x| < 1$.
- We could apply this to any normal matrix U with $\|U\|_\infty < 1$ to get

$$(I - U)^{-1} = \sum_{l=0}^{\infty} U^l.$$

MI is BQP-hard

Goal: Show that any BQP computation V poly-time reducible to an instance A of MI.

Starting point: Let $V = V_m \cdots V_1$ be a BQP circuit on n qubits, $N = 2^n$. Assume WLOG m is power of 2.

Problem: Need to tie matrix inverse with action of V.

Idea:

- Recall Maclaurin series $\frac{1}{1-x} = \sum_{l=0}^{\infty} x^l$ for $|x| < 1$.
- We could apply this to any normal matrix U with $\|U\|_\infty < 1$ to get

$$(I - U)^{-1} = \sum_{l=0}^{\infty} U^l.$$

- What would be great: Normal matrix U acting something like

$$U^k|0^n\rangle \approx V_k \cdots V_1|0^n\rangle.$$

- What would be great: Normal matrix U acting something like

$$U^{k}|0^n\rangle \approx V_k \cdots V_1|0^n\rangle.$$

- What would be great: Normal matrix U acting something like

$$U^k|0^n\rangle \approx V_k \cdots V_1|0^n\rangle.$$

- Define:

$$U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^{\dagger} \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n}),$$

Exercise: Check that U is unitary.

- What would be great: Normal matrix U acting something like

$$U^{k}|0^n\rangle \approx V_k \cdots V_1|0^n\rangle.$$

- Define:

$$U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^{\dagger} \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n}),$$

Exercise: Check that U is unitary.

Exercise: Check that $U^m|0^{\log m}\rangle|0^n\rangle = |m\rangle V|0^n\rangle$.

Implication: Measuring first qubit of second register of $U^m|0^{\log m}\rangle|0^n\rangle$ simulates measuring output qubit of V!

- We could apply this to any normal matrix U with $\|U\|_\infty < 1$ to get $(I - U)^{-1} = \sum_{l=0} U^l$.
- Define $U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^\dagger \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n})$,

- We could apply this to any normal matrix U with $\|U\|_\infty < 1$ to get $(I-U)^{-1} = \sum_{l=0} U^l$.
- Define $U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^\dagger \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n})$,
- Define $A = I - U$. Then,

$$|x\rangle \quad \propto \quad A^{-1}|0^{\log m + n}\rangle$$

- We could apply this to any normal matrix U with $\|U\|_\infty < 1$ to get $(I-U)^{-1} = \sum_{l=0} U^l$.
- Define $U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^\dagger \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n})$,
- Define $A = I - U$. Then,

$$\begin{aligned} |x\rangle &\propto A^{-1}|0^{\log m+n}\rangle \\ &= (I-U)^{-1}|0^{\log m+n}\rangle \end{aligned}$$

- We could apply this to any normal matrix U with $\|U\|_\infty < 1$ to get $(I-U)^{-1} = \sum_{l=0} U^l$.
- Define $U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^\dagger \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n})$,
- Define $A = I - U$. Then,

$$\begin{aligned} |x\rangle &\propto A^{-1}|0^{\log m + n}\rangle \\ &= (I-U)^{-1}|0^{\log m+n}\rangle \\ &\propto \sum_{l=0}^{\infty} U^l |0\rangle^{\log m}|0^n\rangle \end{aligned}$$

- We could apply this to any normal matrix U with $\|U\|_\infty < 1$ to get $(I-U)^{-1} = \sum_{l=0} U^l$.
- Define $U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^\dagger \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n})$,
- Define $A = I - U$. Then,

$$\begin{aligned}
|x\rangle &\propto A^{-1}|0^{\log m+n}\rangle \\
&= (I-U)^{-1}|0^{\log m+n}\rangle \\
&\propto \sum_{l=0}^{\infty} U^l |0\rangle^{\log m}|0^n\rangle \\
&\propto |0\rangle|0^n\rangle + |1\rangle V_1|0^n\rangle + \cdots + |m\rangle V_m \cdots V_1|0^n\rangle.
\end{aligned}$$

- We could apply this to any normal matrix U with $\|U\|_\infty < 1$ to get $(I-U)^{-1} = \sum_{l=0} U^l$.
- Define $U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^\dagger \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n})$,
- Define $A = I - U$. Then,

$$\begin{aligned} |x\rangle &\propto A^{-1}|0^{\log m+n}\rangle \\ &= (I-U)^{-1}|0^{\log m+n}\rangle \\ &\propto \sum_{l=0}^{\infty} U^l |0\rangle^{\log m}|0^n\rangle \\ &\propto |0\rangle|0^n\rangle + |1\rangle V_1|0^n\rangle + \cdots + |m\rangle V_m \cdots V_1 |0^n\rangle. \end{aligned}$$

- Implication:
  - Measuring first register gives $|m\rangle$ with probability $\approx 1/(m+1)$.

- We could apply this to any normal matrix U with $\|U\|_\infty < 1$ to get $(I-U)^{-1} = \sum_{l=0} U^l$.
- Define $U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^\dagger \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n})$,
- Define $A = I - U$. Then,

$$
\begin{aligned}
|x\rangle &\propto A^{-1}|0^{\log m + n}\rangle \\
&= (I-U)^{-1}|0^{\log m+n}\rangle \\
&\propto \sum_{l=0}^{\infty} U^l |0\rangle^{\log m}|0^n\rangle \\
&\propto |0\rangle|0^n\rangle + |1\rangle V_1|0^n\rangle + \cdots + |m\rangle V_m \cdots V_1|0^n\rangle.
\end{aligned}
$$

- Implication:
  - Measuring first register gives $|m\rangle$ with probability $\approx 1/(m+1)$.
  - Postselecting on $|m\rangle$, measuring second register reveals BQP circuit V's output.

- We could apply this to any normal matrix U with $\|U\|_\infty < 1$ to get $(I-U)^{-1} = \sum_{l=0} U^l$.
- Define $U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^\dagger \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n})$,
- Define $A = I - U$. Then,

$$
\begin{aligned}
|x\rangle &\propto A^{-1}|0^{\log m + n}\rangle \\
&= (I-U)^{-1}|0^{\log m+n}\rangle \\
&\propto \sum_{l=0}^{\infty} U^l |0\rangle^{\log m}|0^n\rangle \\
&\propto |0\rangle|0^n\rangle + |1\rangle V_1|0^n\rangle + \cdots + |m\rangle V_m \cdots V_1|0^n\rangle.
\end{aligned}
$$

- Implication:
  - Measuring first register gives $|m\rangle$ with probability $\approx 1/(m+1)$.
  - Postselecting on $|m\rangle$, measuring second register reveals BQP circuit V's output.

Exercise: I cheated slightly on one of the lines above (regarding $|x\rangle$) — where did I cheat?

- We could apply this to any normal matrix U with $\|U\|_\infty < 1$ to get $(I-U)^{-1} = \sum_{l=0} U^l$.
- Define $U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^\dagger \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n})$,
- Define $A = I - U$. Then,

$$\begin{aligned} |x\rangle &\propto A^{-1}|0^{\log m+n}\rangle \\ &= (I-U)^{-1}|0^{\log m+n}\rangle \\ &\propto \sum_{l=0}^{\infty} U^l|0\rangle^{\log m}|0^n\rangle \\ &\propto |0\rangle|0^n\rangle + |1\rangle V_1|0^n\rangle + \cdots + |m\rangle V_m \cdots V_1|0^n\rangle. \end{aligned}$$

- Implication:
  - Measuring first register gives $|m\rangle$ with probability $\approx 1/(m+1)$.
  - Postselecting on $|m\rangle$, measuring second register reveals BQP circuit V's output.

Exercise: I cheated slightly on one of the lines above (regarding $|x\rangle$) — where did I cheat?

Exercise: I cheated less slightly somewhere else on this slide. Where did I make a bigger boo boo?

Final exercises for MI

Construction *almost* works, but for 3 issues to check:

1. A must be $O(1)$-sparse (by def of MI).

   Exercise: Check that U, and thus A, are $O(1)$-sparse.

Final exercises for MI

Construction *almost* works, but for 3 issues to check:

1. A must be $O(1)$-sparse (by def of MI).

   Exercise: Check that U, and thus A, are $O(1)$-sparse.

2. MI needs YES case and NO case thresholds of $2/3$ vs $1/3$ for BQP. The current construction will give $2/(3(m+1))$ vs $1/(3(m+1))$.

   Exercise: Modify the construction to boost the YES/NO thresholds to $2/3$ and $1/3$, respectively.

Final exercises for MI

Construction *almost* works, but for 3 issues to check:

1. A must be $O(1)$-sparse (by def of MI).

   Exercise: Check that U, and thus A, are $O(1)$-sparse.

2. MI needs YES case and NO case thresholds of $2/3$ vs $1/3$ for BQP. The current construction will give $2/(3(m+1))$ vs $1/(3(m+1))$.

   Exercise: Modify the construction to boost the YES/NO thresholds to $2/3$ and $1/3$, respectively.

3. Our current choice of A is not necessarily invertible, since $\|U\|_\infty = 1$. (Maclaurin series does not apply.)

   Exercise: Consider first $A = I - \frac{1}{2}U$. Show that A is invertible and has $\kappa(A) \in O(1)$. Where will this construction nevertheless fail in the analysis?

Final exercises for MI

Construction *almost* works, but for 3 issues to check:

1. A must be $O(1)$-sparse (by def of MI).

   Exercise: Check that U, and thus A, are $O(1)$-sparse.

2. MI needs YES case and NO case thresholds of $2/3$ vs $1/3$ for BQP. The current construction will give $2/(3(m+1))$ vs $1/(3(m+1))$.

   Exercise: Modify the construction to boost the YES/NO thresholds to $2/3$ and $1/3$, respectively.

3. Our current choice of A is not necessarily invertible, since $\|U\|_\infty = 1$. (Maclaurin series does not apply.)

   Exercise: Consider first $A = I - \frac{1}{2}U$. Show that A is invertible and has $\kappa(A) \in O(1)$. Where will this construction nevertheless fail in the analysis?

   Exercise Consider finally $A = I - e^{-1/m}U$. Show that A is invertible, has $\kappa(A) \in O(m) \in \text{polylog}(N)$.

Final exercises for MI

Construction *almost* works, but for 3 issues to check:

1. A must be $O(1)$-sparse (by def of MI).

   Exercise: Check that U, and thus A, are $O(1)$-sparse.

2. MI needs YES case and NO case thresholds of $2/3$ vs $1/3$ for BQP. The current construction will give $2/(3(m+1))$ vs $1/(3(m+1))$.

   Exercise: Modify the construction to boost the YES/NO thresholds to $2/3$ and $1/3$, respectively.

3. Our current choice of A is not necessarily invertible, since $\|U\|_\infty = 1$. (Maclaurin series does not apply.)

   Exercise: Consider first $A = I - \frac{1}{2}U$. Show that A is invertible and has $\kappa(A) \in O(1)$. Where will this construction nevertheless fail in the analysis?

   Exercise Consider finally $A = I - e^{-1/m}U$. Show that A is invertible, has $\kappa(A) \in O(m) \in \text{polylog}(N)$.

4. I cheated again. There is a 4th issue — A must be Hermitian. But I will spare you these details.

Outline

Open season

Observation: For linear systems, given as input Hermitian A, we:

- Showed how to simulate A^{-1} by "manually" inverting eigenvalues, i.e. $\mathbb{A}^{-1} = \sum_i \frac{1}{\lambda_i} |\psi_i\rangle\langle\psi_i|$.
- Used Quantum Phase Estimation (QPE) and post-selection.

Open season

Observation: For linear systems, given as input Hermitian A, we:

- Showed how to simulate A^{-1} by "manually" inverting eigenvalues, i.e. $\mathbb{A}^{-1} = \sum_i \frac{1}{\lambda_i} |\psi_i\rangle\langle\psi_i|$.
- Used Quantum Phase Estimation (QPE) and post-selection.

Question: What other operator functions $f(A)$ can we efficiently simulate?

Open season

Observation: For linear systems, given as input Hermitian A, we:

- Showed how to simulate A^{-1} by "manually" inverting eigenvalues, i.e. $\mathbb{A}^{-1} = \sum_i \frac{1}{\lambda_i}|\psi_i\rangle\langle\psi_i|$.
- Used Quantum Phase Estimation (QPE) and post-selection.

Question: What other operator functions $f(A)$ can we efficiently simulate?

Recall:

- BQP-hardness of MI used Taylor series $f(x) = \frac{1}{1-x} = \sum_{l=0}^{\infty} x^l$ for $|x| < 1$.

Open season

Observation: For linear systems, given as input Hermitian A, we:

- Showed how to simulate A^{-1} by "manually" inverting eigenvalues, i.e. $\mathbb{A}^{-1} = \sum_i \frac{1}{\lambda_i}|\psi_i\rangle\langle\psi_i|$.
- Used Quantum Phase Estimation (QPE) and post-selection.

Question: What other operator functions $f(A)$ can we efficiently simulate?

Recall:

- BQP-hardness of MI used Taylor series $f(x) = \frac{1}{1-x} = \sum_{l=0}^{\infty} x^l$ for $|x| < 1$.
  - Idea: Try to simulate *polynomials* applied to A.

Open season

Observation: For linear systems, given as input Hermitian A, we:

- Showed how to simulate A^{-1} by “manually” inverting eigenvalues, i.e. $\mathbb{A}^{-1} = \sum_i \frac{1}{\lambda_i}|\psi_i\rangle\langle\psi_i|$.
- Used Quantum Phase Estimation (QPE) and post-selection.

Question: What other operator functions $f(A)$ can we efficiently simulate?

Recall:

- BQP-hardness of MI used Taylor series $f(x) = \frac{1}{1-x} = \sum_{l=0}^{\infty} x^l$ for $|x| < 1$.
  - Idea: Try to simulate *polynomials* applied to A.
- A not unitary $\rightarrow$ post-selection still needed.

Remember this?

Quantum Singular Value Transformation (Gilyén, Su, Low, and Wiebe 2019)

- Generalizes Low and Chuang's qubitization approach to *non-square* matrices A
- Given non-square A (embedded as block of unitary U), polynomial p, simulates mapping

$$|\psi\rangle \mapsto p\left(\sqrt{A^\dagger A}\right)|\psi\rangle.$$

- Unified framework for a host of quantum algorithms:
  - Hamiltonian simulation, linear systems, amplitude amplification, quantum machine learning algorithms, and essentially all "quantum matrix linear algebra"
  - [Martyn, Rossi, Tan and Chuang 2021] *iteratively* apply QSVT to simulate Fourier Transform

Challenges

1. How to apply non-unitary (or perhaps not even square) A?
2. Given polynomial p and ability to apply A, how to apply $p(A)$?

Challenges

1. How to apply non-unitary (or perhaps not even square) A?
2. Given polynomial p and ability to apply A, how to apply $p(A)$?

Solutions:

1. Use block encodings of A (more generally, projected unitary encodings).

Challenges

1. How to apply non-unitary (or perhaps not even square) A?
2. Given polynomial p and ability to apply A, how to apply $p(A)$?

Solutions:

1. Use block encodings of A (more generally, projected unitary encodings).
2. Use Quantum Signal Processing (QSP), i.e. qubitization.

Step 1: Block encodings

Recall key step of HHL algorithm ($A = \sum_j \lambda_j |\psi_j\rangle\langle\psi_j|$:

$$\sum_{j=1}^{N} \alpha_j |0\rangle_R |\lambda_j\rangle|\psi_j\rangle \quad \mapsto \quad \sum_{j=1}^{N} \alpha_j \left(\sqrt{1 - \frac{1}{\lambda_j^2 \kappa^2(A)}} |0\rangle + \left(\frac{1}{\lambda_j \kappa(A)} \right) |1\rangle \right)_R |\lambda_j\rangle|\psi_j\rangle.$$

Postselecting on $|1\rangle$ in register R simulated application of A^{-1}, i.e. eigenvector $|\psi_j\rangle$ hit with coefficient λ_j^{-1}.

Step 1: Block encodings

Recall key step of HHL algorithm ($A = \sum_j \lambda_j |\psi_j\rangle\langle\psi_j|$:

$$\sum_{j=1}^{N} \alpha_j |0\rangle_R |\lambda_j\rangle|\psi_j\rangle \quad \mapsto \quad \sum_{j=1}^{N} \alpha_j \left(\sqrt{1 - \frac{1}{\lambda_j^2 \kappa^2(A)}} |0\rangle + \left(\frac{1}{\lambda_j \kappa(A)} \right) |1\rangle \right)_R |\lambda_j\rangle|\psi_j\rangle.$$

Postselecting on $|1\rangle$ in register R simulated application of A^{-1}, i.e. eigenvector $|\psi_j\rangle$ hit with coefficient λ_j^{-1}.

Effective unitary HHL implements (before measuring R):

$$U = \begin{pmatrix} A^{-1} & ? \\ ? & ? \end{pmatrix} = |0\rangle\langle 0|_R \otimes A^{-1} + |0\rangle\langle 1|_R \otimes ? + |1\rangle\langle 0|_R \otimes ? + |1\rangle\langle 1|_R \otimes ?$$

Step 1: Block encodings

Recall key step of HHL algorithm ($A = \sum_j \lambda_j |\psi_j\rangle\langle\psi_j|$:

$$\sum_{j=1}^{N} \alpha_j |0\rangle_R |\lambda_j\rangle|\psi_j\rangle \quad \mapsto \quad \sum_{j=1}^{N} \alpha_j \left(\sqrt{1 - \frac{1}{\lambda_j^2 \kappa^2(A)}} |0\rangle + \left(\frac{1}{\lambda_j \kappa(A)}\right) |1\rangle \right)_R |\lambda_j\rangle|\psi_j\rangle.$$

Postselecting on $|1\rangle$ in register R simulated application of A^{-1}, i.e. eigenvector $|\psi_j\rangle$ hit with coefficient λ_j^{-1}.

Effective unitary HHL implements (before measuring R):

$$U = \begin{pmatrix} A^{-1} & ? \\ ? & ? \end{pmatrix} = |0\rangle\langle 0|_R \otimes A^{-1} + |0\rangle\langle 1|_R \otimes ? + |1\rangle\langle 0|_R \otimes ? + |1\rangle\langle 1|_R \otimes ?$$

So, we may view HHL as doing:

1. Prepare initial state: $|0\rangle_R |b\rangle$

Step 1: Block encodings

Recall key step of HHL algorithm ($A=\sum_j \lambda_j|\psi_j\rangle\langle\psi_j|$:

$$\sum_{j=1}^{N}\alpha_j|0\rangle_R|\lambda_j\rangle|\psi_j\rangle \quad\mapsto\quad \sum_{j=1}^{N}\alpha_j\left(\sqrt{1-\frac{1}{\lambda_j^2\kappa^2(A)}}|0\rangle+\left(\frac{1}{\lambda_j\kappa(A)}\right)|1\rangle\right)_R|\lambda_j\rangle|\psi_j\rangle.$$

Postselecting on $|1\rangle$ in register R simulated application of A^{-1}, i.e. eigenvector $|\psi_j\rangle$ hit with coefficient λ_j^{-1}.

Effective unitary HHL implements (before measuring R):

$$U=\begin{pmatrix} A^{-1} & ? \\ ? & ? \end{pmatrix}=|0\rangle\langle 0|_R\otimes A^{-1}+|0\rangle\langle 1|_R\otimes ?+|1\rangle\langle 0|_R\otimes ?+|1\rangle\langle 1|_R\otimes ?$$

So, we may view HHL as doing:

1. Prepare initial state: $|0\rangle_R|b\rangle$
2. Use QPE to simulate $U|0\rangle_R|b\rangle$.

Step 1: Block encodings

Recall key step of HHL algorithm ($A = \sum_j \lambda_j |\psi_j\rangle\langle\psi_j|$:

$$\sum_{j=1}^{N} \alpha_j |0\rangle_R |\lambda_j\rangle|\psi_j\rangle \quad \mapsto \quad \sum_{j=1}^{N} \alpha_j \left(\sqrt{1 - \frac{1}{\lambda_j^2 \kappa^2(A)}} |0\rangle + \left(\frac{1}{\lambda_j \kappa(A)} \right) |1\rangle \right)_R |\lambda_j\rangle|\psi_j\rangle.$$

Postselecting on $|1\rangle$ in register R simulated application of A^{-1}, i.e. eigenvector $|\psi_j\rangle$ hit with coefficient λ_j^{-1}.

Effective unitary HHL implements (before measuring R):

$$U = \begin{pmatrix} A^{-1} & ? \\ ? & ? \end{pmatrix} = |0\rangle\langle 0|_R \otimes A^{-1} + |0\rangle\langle 1|_R \otimes ? + |1\rangle\langle 0|_R \otimes ? + |1\rangle\langle 1|_R \otimes ?$$

So, we may view HHL as doing:

1. Prepare initial state: $|0\rangle_R |b\rangle$
2. Use QPE to simulate $U|0\rangle_R|b\rangle$.
3. Measure R and postselect on outcome 0:

$$U|0\rangle_R|b\rangle \mapsto (\langle 0|_R \otimes I) U|0\rangle_R|b\rangle \propto A^{-1}|b\rangle.$$

Block encoding

A *block encoding* of matrix A on n qubits is any unitary U s.t.

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} = |0\rangle\langle 0|^{\otimes n} \otimes A + \dots$$

Block encoding

A *block encoding* of matrix A on n qubits is any unitary U s.t.

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} = |0\rangle\langle 0|^{\otimes n} \otimes A + \dots$$

Assumptions:

- We have *efficient* implementation of U.

Block encoding

A *block encoding* of matrix A on n qubits is any unitary U s.t.

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} = |0\rangle\langle 0|^{\otimes n} \otimes A + \ldots$$

Assumptions:

- We have *efficient* implementation of U.
- Probability of post-selecting on $|0\rangle^{\otimes n}$ depends on A and state we apply it to.

Block encoding

A *block encoding* of matrix A on n qubits is any unitary U s.t.

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} = |0\rangle\langle 0|^{\otimes n} \otimes A + \dots$$

Assumptions:

- We have *efficient* implementation of U.
- Probability of post-selecting on $|0\rangle^{\otimes n}$ depends on A and state we apply it to.

More generally:

Projected Unitary Encoding

A *projected unitary encoding* of matrix A on n qubits is (Π_L, U, Π_R) s.t.

$$A = \Pi_L U \Pi_R \quad \text{for projectors } \Pi_L, \Pi_R \text{ and unitary } U.$$

Block encoding

A *block encoding* of matrix A on n qubits is any unitary U s.t.

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} = |0\rangle\langle 0|^{\otimes n} \otimes A + \dots$$

Assumptions:

- We have *efficient* implementation of U.
- Probability of post-selecting on $|0\rangle^{\otimes n}$ depends on A and state we apply it to.

More generally:

Projected Unitary Encoding

A *projected unitary encoding* of matrix A on n qubits is (Π_L, U, Π_R) s.t.

$$A = \Pi_L U \Pi_R \quad \text{for projectors } \Pi_L, \Pi_R \text{ and unitary } U.$$

Exercise. What are Π_L and Π_R in case of block encoding?

Step 2: Quantum Signal Processing

Have projected unitary encoding $A = \Pi_L U \Pi_R$ for efficiently implementable U.

Step 2: Quantum Signal Processing

Have projected unitary encoding $A = \Pi_L U \Pi_R$ for efficiently implementable U.

Given polynomial p, how to obtain projected unitary encoding of $p(A)$?

Step 2: Quantum Signal Processing

- Have projected unitary encoding $A = \Pi_L U \Pi_R$ for efficiently implementable U.
- Want to map:

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \quad \mapsto \quad U = \begin{pmatrix} p(A) & \cdot \\ \cdot & \cdot \end{pmatrix}$$

Step 2: Quantum Signal Processing

- Have projected unitary encoding $A = \Pi_L U \Pi_R$ for efficiently implementable U.
- Want to map:

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \quad \mapsto \quad U = \begin{pmatrix} p(A) & \cdot \\ \cdot & \cdot \end{pmatrix}$$

- Postselecting on $|0\rangle^{\otimes n}$ simulates application of $p(A)$.

Step 2: Quantum Signal Processing

- Have projected unitary encoding $A = \Pi_L U \Pi_R$ for efficiently implementable U.
- Want to map:

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \quad \mapsto \quad U = \begin{pmatrix} p(A) & \cdot \\ \cdot & \cdot \end{pmatrix}$$

- Postselecting on $|0\rangle^{\otimes n}$ simulates application of $p(A)$.

Tool: Quantum Signal Processing (QSP), in two steps:

1. QSP on single qubit system
2. Embed into QSP on larger systems

QSP on single qubit system

- Consider following block encoding of 1×1 Hermitian matrix $[x]$ with $x \in [-1, 1]$:

$$R(x) := \begin{pmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{pmatrix} \in \mathcal{L}(\mathbb{C}^2).$$

QSP on single qubit system

- Consider following block encoding of 1×1 Hermitian matrix $[x]$ with $x \in [-1, 1]$:

$$R(x) := \begin{pmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{pmatrix} \in \mathcal{L}(\mathbb{C}^2).$$

- Want: Given polynomial p, induce map

$$R(x) \quad \mapsto \quad \begin{pmatrix} p(x) & \cdot \\ \cdot & \cdot \end{pmatrix} \in \mathcal{L}(\mathbb{C}^2).$$

QSP on single qubit system

- Consider following block encoding of 1×1 Hermitian matrix $[x]$ with $x \in [-1, 1]$:

$$R(x) := \begin{pmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{pmatrix} \in \mathcal{L}(\mathbb{C}^2).$$

- Want: Given polynomial p, induce map

$$R(x) \quad \mapsto \quad \begin{pmatrix} p(x) & \cdot \\ \cdot & \cdot \end{pmatrix} \in \mathcal{L}(\mathbb{C}^2).$$

- Question:
  - Suppose we can apply R, $R^\dagger$, and $e^{i\theta Z}$ for Pauli Z and any $\theta \in [0, 2\pi]$.
  - For which polynomials p can the mapping above be done, and what is the cost?

QSP on single qubit system

A polynomial $p \in \mathbb{C}[x]$ is odd if all coefficients corresponding to even powers of x are 0.

Alternatively, for all $x \in \mathbb{R}$, $p(-x) = -p(x)$.

QSP on single qubit system

A polynomial $p \in \mathbb{C}[x]$ is odd if all coefficients corresponding to even powers of x are 0.

Alternatively, for all $x \in \mathbb{R}$, $p(-x) = -p(x)$.

QSVT using reflections

Let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

QSP on single qubit system

A polynomial $p \in \mathbb{C}[x]$ is odd if all coefficients corresponding to even powers of x are 0.

Alternatively, for all $x \in \mathbb{R}$, $p(-x) = -p(x)$.

QSVT using reflections

Let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and

QSP on single qubit system

A polynomial $p \in \mathbb{C}[x]$ is odd if all coefficients corresponding to even powers of x are 0.

Alternatively, for all $x \in \mathbb{R}$, $p(-x) = -p(x)$.

QSVT using reflections

Let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

QSP on single qubit system

A polynomial $p \in \mathbb{C}[x]$ is odd if all coefficients corresponding to even powers of x are 0.

Alternatively, for all $x \in \mathbb{R}$, $p(-x) = -p(x)$.

QSVT using reflections

Let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

There exists sequence of d angles $\Phi := (\phi_1, \dots, \phi_d) \in \mathbb{R}^d$ s.t.

$$\begin{pmatrix} p(x) & \cdot \\ \cdot & \cdot \end{pmatrix} =$$

QSP on single qubit system

A polynomial $p \in \mathbb{C}[x]$ is odd if all coefficients corresponding to even powers of x are 0.

Alternatively, for all $x \in \mathbb{R}$, $p(-x) = -p(x)$.

QSVT using reflections

Let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

There exists sequence of d angles $\Phi := (\phi_1, \ldots, \phi_d) \in \mathbb{R}^d$ s.t.

$$\begin{pmatrix} p(x) & \cdot \\ \cdot & \cdot \end{pmatrix} = e^{i\phi_1 Z} R(x) e^{i\phi_2 Z} R(x) \cdots e^{i\phi_d Z} R(x).$$

QSP on single qubit system

A polynomial $p \in \mathbb{C}[x]$ is odd if all coefficients corresponding to even powers of x are 0.

Alternatively, for all $x \in \mathbb{R}$, $p(-x) = -p(x)$.

QSVT using reflections

Let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

There exists sequence of d angles $\Phi := (\phi_1, \ldots, \phi_d) \in \mathbb{R}^d$ s.t.

$$\begin{pmatrix} p(x) & \cdot \\ \cdot & \cdot \end{pmatrix} = e^{i\phi_1 Z} R(x) e^{i\phi_2 Z} R(x) \cdots e^{i\phi_d Z} R(x).$$

Exercise: Why do we need condition (1)?

QSP on single qubit system

A polynomial $p \in \mathbb{C}[x]$ is odd if all coefficients corresponding to even powers of x are 0.

Alternatively, for all $x \in \mathbb{R}$, $p(-x) = -p(x)$.

QSVT using reflections

Let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

There exists sequence of d angles $\Phi := (\phi_1, \ldots, \phi_d) \in \mathbb{R}^d$ s.t.

$$\begin{pmatrix} p(x) & \cdot \\ \cdot & \cdot \end{pmatrix} = e^{i\phi_1 Z} R(x) e^{i\phi_2 Z} R(x) \cdots e^{i\phi_d Z} R(x).$$

Exercise: Why do we need condition (1)?

Exercise: Why do we prefer *low*-degree polynomials?

On to the general case: A acting on n qubits

Singular Value Decomposition

Question: Thus far, mapped real $x \in [-1, 1]$ to $p(x)$. What is high-dimensional analogue of this?

Singular Value Decomposition

Question: Thus far, mapped real $x \in [-1, 1]$ to $p(x)$. What is high-dimensional analogue of this?

Singular Value Decomposition (SVD)

Any matrix $A \in \mathcal{L}(\mathbb{C}^d)$ has singular value decomposition

$$A = \sum_{i=1}^{d} s_i |l_i\rangle\langle r_i|, \text{ for}$$

- $s_i \geq 0$ are *singular values*,
- $\{|l_i\rangle\}$ are orthonormal set of *left singular vectors*,
- $\{|r_i\rangle\}$ are orthonormal set of *right singular vectors*.

Singular Value Decomposition

Question: Thus far, mapped real $x \in [-1, 1]$ to $p(x)$. What is high-dimensional analogue of this?

Singular Value Decomposition (SVD)

Any matrix $A \in \mathcal{L}(\mathbb{C}^d)$ has singular value decomposition

$$A = \sum_{i=1}^{d} s_i |l_i\rangle\langle r_i|, \text{ for}$$

- $s_i \geq 0$ are *singular values*,
- $\{|l_i\rangle\}$ are orthonormal set of *left singular vectors*,
- $\{|r_i\rangle\}$ are orthonormal set of *right singular vectors*.

Goal: Given projected unitary encoding $A = \Pi_L U \Pi_R$ and odd polynomial $p \in \mathbb{C}[x]$, simulate

$$p(A) := \sum_{i=1}^{d} p(s_i) |l_i\rangle\langle r_i|.$$

Singular Value Decomposition

Question: Thus far, mapped real $x \in [-1, 1]$ to $p(x)$. What is high-dimensional analogue of this?

Singular Value Decomposition (SVD)

Any matrix $A \in \mathcal{L}(\mathbb{C}^d)$ has singular value decomposition

$$A = \sum_{i=1}^{d} s_i |l_i\rangle\langle r_i|, \text{ for}$$

- $s_i \geq 0$ are *singular values*,
- $\{|l_i\rangle\}$ are orthonormal set of *left singular vectors*,
- $\{|r_i\rangle\}$ are orthonormal set of *right singular vectors*.

Goal: Given projected unitary encoding $A = \Pi_L U \Pi_R$ and odd polynomial $p \in \mathbb{C}[x]$, simulate

$$p(A) := \sum_{i=1}^{d} p(s_i) |l_i\rangle\langle r_i|.$$

Exercise. Why does this generalize our single-qubit setup? (i.e. previously we had $x \in [-1, 1]$)

QSVT by alternating phase modulation (Gilyén, Su, Low, and Wiebe 2018)

Consider projected unitary encoding $A = \Pi_L U \Pi_R$, and let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

QSVT by alternating phase modulation (Gilyén, Su, Low, and Wiebe 2018)

Consider projected unitary encoding $A = \Pi_L U \Pi_R$, and let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

There exists sequence of d angles $\Phi := (\phi_1, \ldots, \phi_d) \in \mathbb{R}^d$ s.t.

$$p(A) = \Pi_L U_\Phi \Pi_R$$

QSVT by alternating phase modulation (Gilyén, Su, Low, and Wiebe 2018)

Consider projected unitary encoding $A = \Pi_L U \Pi_R$, and let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

There exists sequence of d angles $\Phi := (\phi_1, \dots, \phi_d) \in \mathbb{R}^d$ s.t.

$$p(A) = \Pi_L U_\Phi \Pi_R = \Pi_L \left(e^{i\phi_1(2\Pi_L - I)} U e^{i\phi_2(2\Pi_R - I)} U^\dagger \cdots e^{i\phi_d(2\Pi_L - I)} U \right) \Pi_R.$$

QSVT by alternating phase modulation (Gilyén, Su, Low, and Wiebe 2018)

Consider projected unitary encoding $A = \Pi_L U \Pi_R$, and let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

There exists sequence of d angles $\Phi := (\phi_1, \ldots, \phi_d) \in \mathbb{R}^d$ s.t.

$$p(A) = \Pi_L U_\Phi \Pi_R = \Pi_L \left(e^{i\phi_1(2\Pi_L - I)} U e^{i\phi_2(2\Pi_R - I)} U^\dagger \cdots e^{i\phi_d(2\Pi_L - I)} U \right) \Pi_R.$$

What is the cost of implementing U_Φ?

QSVT by alternating phase modulation (Gilyén, Su, Low, and Wiebe 2018)

Consider projected unitary encoding $A = \Pi_L U \Pi_R$, and let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

There exists sequence of d angles $\Phi := (\phi_1, \ldots, \phi_d) \in \mathbb{R}^d$ s.t.

$$p(A) = \Pi_L U_\Phi \Pi_R = \Pi_L \left(e^{i\phi_1(2\Pi_L - I)} U e^{i\phi_2(2\Pi_R - I)} U^\dagger \cdots e^{i\phi_d(2\Pi_L - I)} U \right) \Pi_R.$$

What is the cost of implementing U_Φ?

- $O(m)$ uses of U and $U^\dagger$

QSVT by alternating phase modulation (Gilyén, Su, Low, and Wiebe 2018)

Consider projected unitary encoding $A = \Pi_L U \Pi_R$, and let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

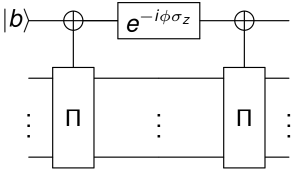
1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

There exists sequence of d angles $\Phi := (\phi_1, \ldots, \phi_d) \in \mathbb{R}^d$ s.t.

$$p(A) = \Pi_L U_\Phi \Pi_R = \Pi_L \left(e^{i\phi_1(2\Pi_L - I)} U e^{i\phi_2(2\Pi_R - I)} U^\dagger \cdots e^{i\phi_d(2\Pi_L - I)} U \right) \Pi_R.$$

What is the cost of implementing U_Φ?

- $O(m)$ uses of U and $U^\dagger$
- $O(m)$ uses of following circuit which implements map $|b\rangle\langle b| \otimes e^{(-1)^b i\phi(2\Pi - I)}$:



Sanity check

We said that for projected unitary encoding $A = \Pi_L U \Pi_R$ and polynomial $p \in \mathbb{C}[x]$,

$$p(A) = \Pi_L \left(e^{i\phi_1(2\Pi_L - I)} U e^{i\phi_2(2\Pi_R - I)} U^\dagger \cdots e^{i\phi_d(2\Pi_L - I)} U \right) \Pi_R.$$

Exercise: What happens if $A = U$, i.e. $\Pi_L = \Pi_R = I$, meaning no block encoding necessary?

Sanity check

We said that for projected unitary encoding $A = \Pi_L U \Pi_R$ and polynomial $p \in \mathbb{C}[x]$,

$$p(A) = \Pi_L \left(e^{i\phi_1(2\Pi_L - I)} U e^{i\phi_2(2\Pi_R - I)} U^\dagger \cdots e^{i\phi_d(2\Pi_L - I)} U \right) \Pi_R.$$

Exercise: What happens if $A = U$, i.e. $\Pi_L = \Pi_R = I$, meaning no block encoding necessary?

Exercise: Aren't you forgetting to ask a very important question? (Hint: The sequence $(\phi_1, \ldots, \phi_d)$.)

Proof intuition

Key insight: Can decompose all operators into direct sum of 1- and 2-dimensional subspaces.

Proof intuition

Key insight: Can decompose all operators into direct sum of 1- and 2-dimensional subspaces.

Recall: Given $A = \Pi_L U \Pi_R = \sum_{i=1}^{d} s_i |l_i\rangle\langle r_i|$. Order singular values:

$$\underbrace{\underbrace{s_1 \geq s_2 \geq \cdots \geq s_k}_{=1} \geq \underbrace{s_{k+1} \geq \cdots \geq s_r}_{0<\cdot<1}}_{r=\mathrm{rank}(A)} \geq \underbrace{s_{r+1} \geq \cdots \geq s_d}_{=0}$$

Proof intuition

Key insight: Can decompose all operators into direct sum of 1- and 2-dimensional subspaces.

Recall: Given $A = \Pi_L U \Pi_R = \sum_{i=1}^{d} s_i |l_i\rangle\langle r_i|$. Order singular values:

$$\underbrace{\underbrace{s_1 \geq s_2 \geq \cdots \geq s_k}_{=1} \geq \underbrace{s_{k+1} \geq \cdots \geq s_r}_{0<\cdot<1}}_{r=\operatorname{rank}(A)} \geq \underbrace{s_{r+1} \geq \cdots \geq s_d}_{=0}$$

Theorem (Invariant subspaces)

$$U = \bigoplus_{i\in[k]} [s_i]_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[r]\setminus[k]} \begin{bmatrix} s_i & \sqrt{1-s_i^2} \\ \sqrt{1-s_i^2} & -s_i \end{bmatrix}_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]_{\tilde{\mathcal{H}}_i^R}^{\mathcal{H}_i^R} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]_{\tilde{\mathcal{H}}_i^L}^{\mathcal{H}_i^L} \oplus [\cdot]_{\tilde{\mathcal{H}}_\perp}^{\mathcal{H}_\perp}$$

$$e^{i\phi(2\Pi - I)} = \bigoplus_{i\in[k]} \left[e^{i\phi}\right]_{\mathcal{H}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[r]\setminus[k]} \begin{bmatrix} e^{i\phi} & 0 \\ 0 & e^{-i\phi} \end{bmatrix}_{\mathcal{H}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[d]\setminus[r]} \left[e^{i\phi}\right]_{\mathcal{H}_i^R}^{\mathcal{H}_i^R} \oplus \bigoplus_{i\in[d]\setminus[r]} \left[e^{-i\phi}\right]_{\mathcal{H}_i^L}^{\mathcal{H}_i^L} \oplus [\cdot]_{\mathcal{H}_\perp}^{\mathcal{H}_\perp},$$

Proof intuition

$$U = \bigoplus_{i\in[k]} [s_i]_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[r]\setminus[k]} \begin{bmatrix} s_i & \sqrt{1-s_i^2} \\ \sqrt{1-s_i^2} & -s_i \end{bmatrix}_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]_{\tilde{\mathcal{H}}_i^R}^{\mathcal{H}_i^R} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]_{\tilde{\mathcal{H}}_i^L}^{\mathcal{H}_i^L} \oplus [\,\cdot\,]_{\tilde{\mathcal{H}}_\perp}^{\mathcal{H}_\perp}$$

Recall: k largest index with $s_k = 1$, r largest index with $s_r > 0$ (i.e. $r = \mathrm{rank}(A)$ for $A = \Pi_L U \Pi_R$).

Proof intuition

$$U = \bigoplus_{i\in[k]} [s_i]_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[r]\setminus[k]} \begin{bmatrix} s_i & \sqrt{1-s_i^2} \\ \sqrt{1-s_i^2} & -s_i \end{bmatrix}_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]_{\tilde{\mathcal{H}}_i^R}^{\mathcal{H}_i^R} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]_{\tilde{\mathcal{H}}_i^L}^{\mathcal{H}_i^L} \oplus [\,\cdot\,]_{\tilde{\mathcal{H}}_\perp}^{\mathcal{H}_\perp}$$

Recall: k largest index with $s_k = 1$, r largest index with $s_r > 0$ (i.e. $r = \mathrm{rank}\,(A)$ for $A = \Pi_L U \Pi_R$).

Question: What are the spaces $\mathcal{H}_i$ and $\tilde{\mathcal{H}}_i$?

Proof intuition

$$U = \bigoplus_{i\in[k]} [s_i]^{\mathcal{H}_i}_{\tilde{\mathcal{H}}_i} \oplus \bigoplus_{i\in[r]\setminus[k]} \begin{bmatrix} s_i & \sqrt{1-s_i^2} \\ \sqrt{1-s_i^2} & -s_i \end{bmatrix}^{\mathcal{H}_i}_{\tilde{\mathcal{H}}_i} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]^{\mathcal{H}_i^R}_{\tilde{\mathcal{H}}_i^R} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]^{\mathcal{H}_i^L}_{\tilde{\mathcal{H}}_i^L} \oplus [\,\cdot\,]^{\mathcal{H}_\perp}_{\tilde{\mathcal{H}}_\perp}$$

Recall: k largest index with $s_k = 1$, r largest index with $s_r > 0$ (i.e. $r = \operatorname{rank}(A)$ for $A = \Pi_L U \Pi_R$).

Question: What are the spaces $\mathcal{H}_i$ and $\tilde{\mathcal{H}}_i$?

$$1 \leq i \leq k \qquad \mathcal{H}_i := \operatorname{Span}(|r_i\rangle) \qquad \tilde{\mathcal{H}}_i := \operatorname{Span}(|l_i\rangle),$$

$$k < i \leq r \qquad \mathcal{H}_i := \operatorname{Span}\left(|r_i\rangle, |r_i^\perp\rangle\right) \qquad |r_i^\perp\rangle := \frac{(I-\Pi)U^\dagger|l_i\rangle}{\|(I-\Pi)U^\dagger|l_i\rangle\|} = \frac{(I-\Pi)U^\dagger|l_i\rangle}{\sqrt{1-s_i^2}},$$

$$\tilde{\mathcal{H}}_i := \operatorname{Span}\left(|l_i\rangle, |l_i^\perp\rangle\right) \qquad |l_i^\perp\rangle := \frac{(I-\widetilde{\Pi})U|r_i\rangle}{\left\|(I-\widetilde{\Pi})U|r_i\rangle\right\|} = \frac{(I-\widetilde{\Pi})U|r_i\rangle}{\sqrt{1-s_i^2}}$$

Proof intuition

Theorem (Invariant subspaces)

$$U = \bigoplus_{i\in[k]} [s_i]^{\mathcal{H}_i}_{\tilde{\mathcal{H}}_i} \oplus \bigoplus_{i\in[r]\setminus[k]} \begin{bmatrix} s_i & \sqrt{1-s_i^2} \\ \sqrt{1-s_i^2} & -s_i \end{bmatrix}^{\mathcal{H}_i}_{\tilde{\mathcal{H}}_i} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]^{\mathcal{H}_i^R}_{\tilde{\mathcal{H}}_i^R} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]^{\mathcal{H}_i^L}_{\tilde{\mathcal{H}}_i^L} \oplus [\cdot]^{\mathcal{H}_\perp}_{\tilde{\mathcal{H}}_\perp}$$

$$e^{i\phi(2\Pi - I)} = \bigoplus_{i\in[k]} \left[e^{i\phi}\right]^{\mathcal{H}_i}_{\mathcal{H}_i} \oplus \bigoplus_{i\in[r]\setminus[k]} \begin{bmatrix} e^{i\phi} & 0 \\ 0 & e^{-i\phi} \end{bmatrix}^{\mathcal{H}_i}_{\mathcal{H}_i} \oplus \bigoplus_{i\in[d]\setminus[r]} \left[e^{i\phi}\right]^{\mathcal{H}_i^R}_{\mathcal{H}_i^R} \oplus \bigoplus_{i\in[d]\setminus[r]} \left[e^{-i\phi}\right]^{\mathcal{H}_i^L}_{\mathcal{H}_i^L} \oplus [\cdot]^{\mathcal{H}_\perp}_{\mathcal{H}_\perp},$$

Proof intuition

Theorem (Invariant subspaces)

$$U = \bigoplus_{i\in[k]} [s_i]_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[r]\setminus[k]} \begin{bmatrix} s_i & \sqrt{1-s_i^2} \\ \sqrt{1-s_i^2} & -s_i \end{bmatrix}_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]_{\tilde{\mathcal{H}}_i^R}^{\mathcal{H}_i^R} \oplus \bigoplus_{i\in[d]\setminus[r]} [1]_{\tilde{\mathcal{H}}_i^L}^{\mathcal{H}_i^L} \oplus [\cdot]_{\tilde{\mathcal{H}}_\perp}^{\mathcal{H}_\perp}$$

$$e^{i\phi(2\Pi-I)} = \bigoplus_{i\in[k]} \left[e^{i\phi}\right]_{\mathcal{H}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[r]\setminus[k]} \begin{bmatrix} e^{i\phi} & 0 \\ 0 & e^{-i\phi} \end{bmatrix}_{\mathcal{H}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[d]\setminus[r]} \left[e^{i\phi}\right]_{\mathcal{H}_i^R}^{\mathcal{H}_i^R} \oplus \bigoplus_{i\in[d]\setminus[r]} \left[e^{-i\phi}\right]_{\mathcal{H}_i^L}^{\mathcal{H}_i^L} \oplus [\cdot]_{\mathcal{H}_\perp}^{\mathcal{H}_\perp},$$

Alternating these two yields:

$$U_\Phi = \bigoplus_{i\in[k]} [P(s_i)]_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[r]\setminus[k]} \begin{bmatrix} P(s_i) & \cdot \\ \cdot & \cdot \end{bmatrix}_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[d]\setminus[r]} \left[e^{i\phi_0}\right]_{\tilde{\mathcal{H}}_i^R}^{\mathcal{H}_i^R} \oplus \bigoplus_{i\in[d]\setminus[r]} \left[e^{-i\phi_0}\right]_{\tilde{\mathcal{H}}_i^L}^{\mathcal{H}_i^L} \oplus [\cdot]_{\tilde{\mathcal{H}}_\perp}^{\mathcal{H}_\perp}.$$

$$\Pi_L U_\Phi \Pi_R = \bigoplus_{i\in[k]} [P(s_i)]_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[r]\setminus[k]} \begin{bmatrix} P(s_i) & 0 \\ 0 & 0 \end{bmatrix}_{\tilde{\mathcal{H}}_i}^{\mathcal{H}_i} \oplus \bigoplus_{i\in[d]\setminus[r]} [0]_{\tilde{\mathcal{H}}_i^R}^{\mathcal{H}_i^R} \oplus \bigoplus_{i\in[d]\setminus[r]} [0]_{\tilde{\mathcal{H}}_i^L}^{\mathcal{H}_i^L} \oplus [0]_{\tilde{\mathcal{H}}_\perp}^{\mathcal{H}_\perp} = \sum_{i=1}^{d} P(s_i)|l_i\rangle\langle r_i|.$$

QSVT by alternating phase modulation

Consider projected unitary encoding $A = \Pi_L U \Pi_R$, and let $p \in \mathbb{C}[x]$ be odd polynomial of degree d, s.t.

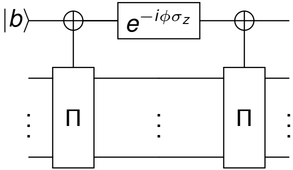
1. for all $x \in [-1, 1]$, $|p(x)| \leq 1$, and
2. for all $x \in [-\infty, -1] \cup [1, \infty]$, $|p(x)| \geq 1$.

There exists sequence of d angles $\Phi := (\phi_1, \ldots, \phi_d) \in \mathbb{R}^d$ s.t.

$$p(A) = \Pi_L U_\Phi \Pi_R = \Pi_L \left(e^{i\phi_1(2\Pi_L - I)} U e^{i\phi_2(2\Pi_R - I)} U^\dagger \cdots e^{i\phi_d(2\Pi_L - I)} U \right) \Pi_R.$$

What is the cost of implementing U_Φ?

- $O(m)$ uses of U and $U^\dagger$
- $O(m)$ uses of following circuit which implements map $|b\rangle\langle b| \otimes e^{(-1)^b i\phi(2\Pi - I)}$:



Example 1: Linear systems via QSVT

Goal: Given projected unitary encoding of $A = \Pi_L U \Pi_R$, want to apply Moore-Penrose pseudoinverse A^+.

Example 1: Linear systems via QSVT

Goal: Given projected unitary encoding of $A = \Pi_L U \Pi_R$, want to apply Moore-Penrose pseudoinverse A^+.

Recall: For SVD $A = \sum_{i=1}^{d} s_i |l_i\rangle\langle r_i| \in \mathcal{L}(\mathbb{C}^d)$, pseudoinverse is

$$A^+ := \sum_{i=1}^{d} \frac{1}{s_i} |r_i\rangle\langle l_i| \quad \text{(for clarity, we invert only } s_i > 0\text{).}$$

Algorithm sketch

1. Pick singular value cutoff $\delta > 0$, i.e. we will invert only $s_i \geq \delta$.

Example 1: Linear systems via QSVT

Goal: Given projected unitary encoding of $A = \Pi_L U \Pi_R$, want to apply Moore-Penrose pseudoinverse A^+.

Recall: For SVD $A = \sum_{i=1}^{d} s_i |l_i\rangle\langle r_i| \in \mathcal{L}(\mathbb{C}^d)$, pseudoinverse is

$$A^+ := \sum_{i=1}^{d} \frac{1}{s_i} |r_i\rangle\langle l_i| \quad \text{(for clarity, we invert only } s_i > 0\text{).}$$

Algorithm sketch

1. Pick singular value cutoff $\delta > 0$, i.e. we will invert only $s_i \geq \delta$.
2. Design low-degree polynomial p which ϵ-approximates $f(x) = \frac{1}{x}$.
   - Degree $d \in O\left(\frac{1}{\delta}\log(\frac{1}{\epsilon})\right)$ suffices.

Example 1: Linear systems via QSVT

Goal: Given projected unitary encoding of $A = \Pi_L U \Pi_R$, want to apply Moore-Penrose pseudoinverse A^+.

Recall: For SVD $A = \sum_{i=1}^{d} s_i |l_i\rangle\langle r_i| \in \mathcal{L}(\mathbb{C}^d)$, pseudoinverse is

$$A^+ := \sum_{i=1}^{d} \frac{1}{s_i} |r_i\rangle\langle l_i| \quad \text{(for clarity, we invert only } s_i > 0\text{)}.$$

Algorithm sketch

1. Pick singular value cutoff $\delta > 0$, i.e. we will invert only $s_i \geq \delta$.
2. Design low-degree polynomial p which ϵ-approximates $f(x) = \frac{1}{x}$.
   - Degree $d \in O\left(\frac{1}{\delta}\log(\frac{1}{\epsilon})\right)$ suffices.
3. Apply QSVT to compute $p(A) \approx A^+$.
   - Costs $O(d)$ uses of U, $U^\dagger$, and Controlled-Π gates

Example 1: Linear systems via QSVT

Goal: Given projected unitary encoding of $A = \Pi_L U \Pi_R$, want to apply Moore-Penrose pseudoinverse A^+.

Recall: For SVD $A = \sum_{i=1}^d s_i |l_i\rangle\langle r_i| \in \mathcal{L}(\mathbb{C}^d)$, pseudoinverse is

$$A^+ := \sum_{i=1}^{d} \frac{1}{s_i} |r_i\rangle\langle l_i| \quad \text{(for clarity, we invert only } s_i > 0\text{)}.$$

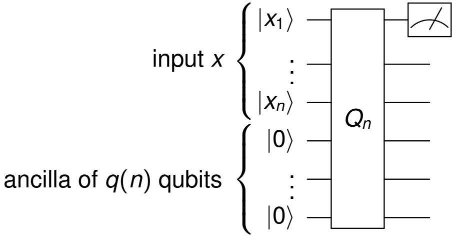
Algorithm sketch

1. Pick singular value cutoff $\delta > 0$, i.e. we will invert only $s_i \geq \delta$.
2. Design low-degree polynomial p which ϵ-approximates $f(x) = \frac{1}{x}$.
   - Degree $d \in O\left(\frac{1}{\delta}\log(\frac{1}{\epsilon})\right)$ suffices.
3. Apply QSVT to compute $p(A) \approx A^+$.
   - Costs $O(d)$ uses of U, $U^\dagger$, and Controlled-Π gates
   - More generally, cost is $O(\|A\|_{\mathrm{F}} / \delta)$ (i.e. if we don't assume $\|A\| \leq 1$).

Example 2: Amplitude Amplification via QSVT

Goal:

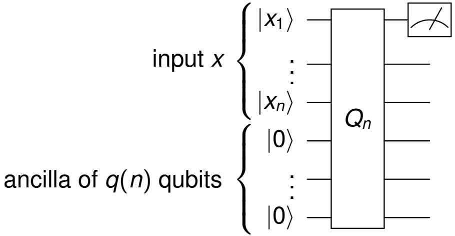
- Given BQP circuit U which outputs 1 with probability $\geq p$.
- Compile new circuit U' which outputs 1 with probability ≈ 1.



Example 2: Amplitude Amplification via QSVT

Goal:

- Given BQP circuit U which outputs 1 with probability $\geq p$.
- Compile new circuit U' which outputs 1 with probability ≈ 1.



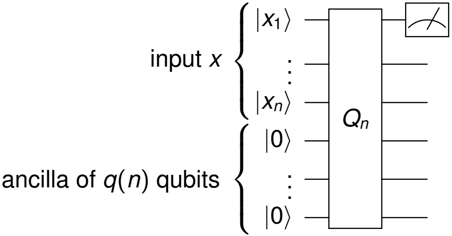
How to setup QSVT:

- Set $\Pi_R := |x\rangle\langle x| \otimes |0\rangle\langle 0|^{\otimes q(n)}$ and $\Pi_L := |1\rangle\langle 1| \otimes I$.

Example 2: Amplitude Amplification via QSVT

Goal:

- Given BQP circuit U which outputs 1 with probability $\geq p$.
- Compile new circuit U' which outputs 1 with probability ≈ 1.



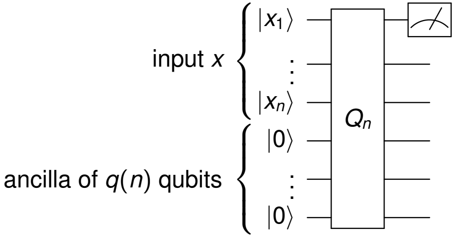
How to setup QSVT:

- Set $\Pi_R := |x\rangle\langle x| \otimes |0\rangle\langle 0|^{\otimes q(n)}$ and $\Pi_L := |1\rangle\langle 1| \otimes I$.
- Then, $A = \Pi_L U \Pi_R$ is rank 1 with singular value the square root of acceptance probability of U.

Example 2: Amplitude Amplification via QSVT

Goal:

- Given BQP circuit U which outputs 1 with probability $\geq p$.
- Compile new circuit U' which outputs 1 with probability ≈ 1.



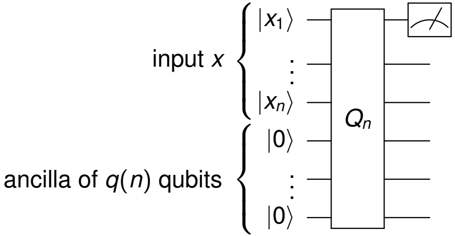
How to setup QSVT:

- Set $\Pi_R := |x\rangle\langle x| \otimes |0\rangle\langle 0|^{\otimes q(n)}$ and $\Pi_L := |1\rangle\langle 1| \otimes I$.
- Then, $A = \Pi_L U \Pi_R$ is rank 1 with singular value the square root of acceptance probability of U.
- Pick a polynomial p which is $\epsilon/2$-close to 1 on $[\sqrt{p}, 1]$.

Example 2: Amplitude Amplification via QSVT

Goal:

- Given BQP circuit U which outputs 1 with probability $\geq p$.
- Compile new circuit U' which outputs 1 with probability ≈ 1.



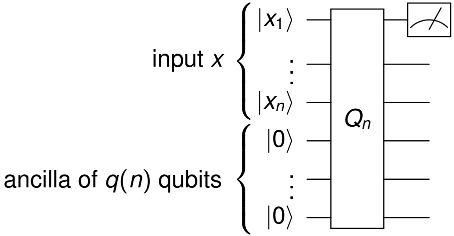
How to setup QSVT:

- Set $\Pi_R := |x\rangle\langle x| \otimes |0\rangle\langle 0|^{\otimes q(n)}$ and $\Pi_L := |1\rangle\langle 1| \otimes I$.
- Then, $A = \Pi_L U \Pi_R$ is rank 1 with singular value the square root of acceptance probability of U.
- Pick a polynomial p which is $\epsilon/2$-close to 1 on $[\sqrt{p}, 1]$.
- Apply QSVT with p to A to get $p(A) = \Pi_L U' \Pi_R$ s..t U' accepts with probability $\geq 1 - \epsilon$.

Example 2: Amplitude Amplification via QSVT

Goal:

- Given BQP circuit U which outputs 1 with probability $\geq p$.
- Compile new circuit U' which outputs 1 with probability ≈ 1.



How to setup QSVT:

- Set $\Pi_R := |x\rangle\langle x| \otimes |0\rangle\langle 0|^{\otimes q(n)}$ and $\Pi_L := |1\rangle\langle 1| \otimes I$.
- Then, $A = \Pi_L U \Pi_R$ is rank 1 with singular value the square root of acceptance probability of U.
- Pick a polynomial p which is $\epsilon/2$-close to 1 on $[\sqrt{p}, 1]$.
- Apply QSVT with p to A to get $p(A) = \Pi_L U' \Pi_R$ s..t U' accepts with probability $\geq 1 - \epsilon$.
- Suffices to choose p of degree $O\left(\frac{1}{\sqrt{p}} \log(\frac{1}{\epsilon})\right)$.

Outline

The story of dequantization

Quantum recommendation systems (Kerenidis, Prakash, 2016)

- Recommendation system (used by, e.g., Netflix):
  - Use ratings of n products by m users to provide personalized recommendations to users
  - Modelled as $m \times n$ preference matrix, assumed to have good rank-k approximation

The story of dequantization

Quantum recommendation systems (Kerenidis, Prakash, 2016)

- Recommendation system (used by, e.g., Netflix):
  - Use ratings of n products by m users to provide personalized recommendations to users
  - Modelled as $m \times n$ preference matrix, assumed to have good rank-k approximation
- Quantum machine learning algorithm which runs in time $\text{poly}(k, \log(mn))$.

The story of dequantization

Conversation between Scott Aaronson and his 18-year old undergrad student, Ewin Tang:

- Scott: I like this paper of Iordanis and Anupam. Can you prove it can't be simulated classically?

The story of dequantization

Conversation between Scott Aaronson and his 18-year old undergrad student, Ewin Tang:

- Scott: I like this paper of Iordanis and Anupam. Can you prove it can't be simulated classically?
- Ewin: What do you mean?

The story of dequantization

Conversation between Scott Aaronson and his 18-year old undergrad student, Ewin Tang:

- Scott: I like this paper of Iordanis and Anupam. Can you prove it can't be simulated classically?
- Ewin: What do you mean?
- Scott: Maybe it's BQP-complete like the linear systems problem?

The story of dequantization

Conversation between Scott Aaronson and his 18-year old undergrad student, Ewin Tang:

- Scott: I like this paper of Iordanis and Anupam. Can you prove it can't be simulated classically?
- Ewin: What do you mean?
- Scott: Maybe it's BQP-complete like the linear systems problem?

A quantum-inspired classical algorithm for recommendation systems (Tang, STOC 2019):

We give a classical analogue to Kerenidis and Prakash's quantum recommendation system, previously believed to be one of the strongest candidates for provably exponential speedups in quantum machine learning. Our main result is an algorithm that, given an $m \times n$ matrix in a data structure supporting certain ℓ^2-norm sampling operations, outputs an ℓ^2-norm sample from a rank-k approximation of that matrix in time $O(\mathrm{poly}(k)\log(mn))$, only polynomially slower than the quantum algorithm. As a consequence, Kerenidis and Prakash's algorithm does not in fact give an exponential speedup over classical algorithms. Further, under strong input assumptions, the classical recommendation system resulting from our algorithm produces recommendations exponentially faster than previous classical systems, which run in time linear in m and n.

The main insight of this work is the use of simple routines to manipulate ℓ^2-norm sampling distributions, which play the role of quantum superpositions in the classical setting. This correspondence indicates a potentially fruitful framework for formally comparing quantum machine learning algorithms to classical machine learning algorithms.

There's more...

There's more...

Dequantizing QSVT in low-rank settings (Chia, Gilyén, Li, Lin, Tang, Wang, STOC 2020)

- Idea: l^2-norm sampling approximates matrix products in time independent of dimension

There's more...

Dequantizing QSVT in low-rank settings (Chia, Gilyén, Li, Lin, Tang, Wang, STOC 2020)

- Idea: l^2-norm sampling approximates matrix products in time independent of dimension
- Dequantizes many quantum machine learning algorithms, including:
  - recommendation systems
  - principal component analysis
  - low-rank regression
  - supervised clustering
  - support vector machines

There's more...

Dequantizing QSVT in low-rank settings (Chia, Gilyén, Li, Lin, Tang, Wang, STOC 2020)

- Idea: l^2-norm sampling approximates matrix products in time independent of dimension
- Dequantizes many quantum machine learning algorithms, including:
  - recommendation systems
  - principal component analysis
  - low-rank regression
  - supervised clustering
  - support vector machines

So, what does this mean?

There's more...

Dequantizing QSVT in low-rank settings (Chia, Gilyén, Li, Lin, Tang, Wang, STOC 2020)

- Idea: l^2-norm sampling approximates matrix products in time independent of dimension
- Dequantizes many quantum machine learning algorithms, including:
  - recommendation systems
  - principal component analysis
  - low-rank regression
  - supervised clustering
  - support vector machines

So, what does this mean?

Tang's dequantization of quantum recommender systems

The quantum recommendation system algorithm relies on the following classical data structure.

Tang's dequantization of quantum recommender systems

The quantum recommendation system algorithm relies on the following classical data structure.

Lemma ((Kerenidis, Prakash 2017), as stated in (Tang 2019))

$\exists$ data structure storing $v \in \mathbb{R}^n$ with w nonzero entries in $O(w \log(n))$ space, which supports:

- Reading and updating an entry of v in $O(\log n)$ time;
- Finding $\|v\|^2$ in $O(1)$ time;
- Sampling from distribution $v_i^2 / \|v\|^2$ in $O(\log n)$ time.

Tang's dequantization of quantum recommender systems

The quantum recommendation system algorithm relies on the following classical data structure.

Lemma ((Kerenidis, Prakash 2017), as stated in (Tang 2019))

$\exists$ data structure storing $v \in \mathbb{R}^n$ with w nonzero entries in $O(w \log(n))$ space, which supports:

- Reading and updating an entry of v in $O(\log n)$ time;
- Finding $\|v\|^2$ in $O(1)$ time;
- Sampling from distribution $v_i^2 / \|v\|^2$ in $O(\log n)$ time.

Observation: Precisely conditions for randomized linear algebra techniques! (Frieze, Kannan, Vempala 2004)

Tang's dequantization of quantum recommender systems

The quantum recommendation system algorithm relies on the following classical data structure.

Lemma ((Kerenidis, Prakash 2017), as stated in (Tang 2019))

$\exists$ data structure storing $v \in \mathbb{R}^n$ with w nonzero entries in $O(w\log(n))$ space, which supports:

- Reading and updating an entry of v in $O(\log n)$ time;
- Finding $\|v\|^2$ in $O(1)$ time;
- Sampling from distribution $v_i^2 / \|v\|^2$ in $O(\log n)$ time.

Observation: Precisely conditions for randomized linear algebra techniques! (Frieze, Kannan, Vempala 2004)

Upshot:

- Any quantum algorithm with input encoded as above can, in principle, be attacked via dequantization.

Tang's dequantization of quantum recommender systems

The quantum recommendation system algorithm relies on the following classical data structure.

Lemma ((Kerenidis, Prakash 2017), as stated in (Tang 2019))

$\exists$ data structure storing $v \in \mathbb{R}^n$ with w nonzero entries in $O(w \log(n))$ space, which supports:

- Reading and updating an entry of v in $O(\log n)$ time;
- Finding $\|v\|^2$ in $O(1)$ time;
- Sampling from distribution $v_i^2 / \|v\|^2$ in $O(\log n)$ time.

Observation: Precisely conditions for randomized linear algebra techniques! (Frieze, Kannan, Vempala 2004)

Upshot:

- Any quantum algorithm with input encoded as above can, in principle, be attacked via dequantization.
- Classical dequantized algorithms still typically polynomially slower than quantum algorithms

Tang's dequantization of quantum recommender systems

The quantum recommendation system algorithm relies on the following classical data structure.

Lemma ((Kerenidis, Prakash 2017), as stated in (Tang 2019))

$\exists$ data structure storing $v \in \mathbb{R}^n$ with w nonzero entries in $O(w \log(n))$ space, which supports:

- Reading and updating an entry of v in $O(\log n)$ time;
- Finding $\|v\|^2$ in $O(1)$ time;
- Sampling from distribution $v_i^2 / \|v\|^2$ in $O(\log n)$ time.

Observation: Precisely conditions for randomized linear algebra techniques! (Frieze, Kannan, Vempala 2004)

Upshot:

- Any quantum algorithm with input encoded as above can, in principle, be attacked via dequantization.
- Classical dequantized algorithms still typically polynomially slower than quantum algorithms
- One has to be careful in assumptions about how the input is specified!

Outline

Guided local Hamiltonian problem (GLH) [G, Le Gall 2022]

- Input: sparse Hamiltonian H on n qubits, $\alpha < \beta$, samplable $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$
- Promise: $\lambda_{\min}(H) \leq \alpha$ or $\lambda_{\min}(H) \geq \beta$, $\| \Pi_H |\psi\rangle \|_2 \geq \delta$
- Output: Decide whether $\lambda_{\min}(H) \leq \alpha$ or $\lambda_{\min}(H) \geq \beta$

Guided local Hamiltonian problem (GLH) [G, Le Gall 2022]

- Input: sparse Hamiltonian H on n qubits, $\alpha < \beta$, samplable $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$
- Promise: $\lambda_{\min}(H) \leq \alpha$ or $\lambda_{\min}(H) \geq \beta$, $\| \Pi_H |\psi\rangle \|_2 \geq \delta$
- Output: Decide whether $\lambda_{\min}(H) \leq \alpha$ or $\lambda_{\min}(H) \geq \beta$

ζ-samplable state for $\zeta \in [0, 1)$

Have ζ-sampling-access to $|\psi\rangle \in \mathbb{C}^{2^n}$ if all three hold:

- (query access) For any $i \in [2^n]$, can compute $\psi_i \in \mathbb{C}$ in $\mathrm{poly}(n)$ classical time
- (sampling access) Can sample in $\mathrm{poly}(n)$ classical time from distribution $p\colon [2^n] \to [0, 1]$ such that

$$\forall j \in [2^n] \qquad p(j) \in \left[(1-\zeta)\frac{|\psi_j|^2}{\| |\psi\rangle \|^2}, (1+\zeta)\frac{|\psi_j|^2}{\| |\psi\rangle \|^2} \right]$$

- (norm approximation) Have m s.t. $|m - \| |\psi\rangle \| \,| \leq \zeta \, \| |\psi\rangle \|$.

Note: When $\zeta = 0$, recover [Tang 2019]'s definition from dequantization of recommender systems

$n = \#$ of qubits

Theorem: GLH "tractable" in $O(1)$-precision setting

$\forall$ constants $\delta, \alpha, \beta \in (0,1]$ and $k \in O(\log n)$, GLH classically solvable in $\mathrm{poly}(n)$ time with probability $1 - 2^{-n}$.

$n = \#$ of qubits

Theorem: GLH "tractable" in $O(1)$-precision setting

$\forall$ constants $\delta, \alpha, \beta \in (0,1]$ and $k \in O(\log n)$, GLH classically solvable in $\mathrm{poly}(n)$ time with probability $1 - 2^{-n}$.

$$\Uparrow$$

Theorem (informal)

The sparse "Guided Singular Value Estimation" problem is efficiently solvable to $O(1)$ precision.

$$\Uparrow$$

choose constant-degree polynomial P in QSVT to "process" singular values

$\rightarrow$ possible in $O(1)$-precision setting

Theorem (informal)

The sparse Quantum Singular Value Transform (QSVT) can be "dequantized" for $O(1)$ precision.

Dequantizing the QSVT in the sparse setting

Singular Value Transform (SVT)

Input: (1) query-access to s-sparse matrix $A \in \mathbb{C}^{M \times N}$ with $\|A\| \leq 1$
(2) query-access to $u \in \mathbb{C}^N$ s.t. $\|u\| \leq 1$
(3) ζ-samplable $v \in \mathbb{C}^N$ s.t. $\|v\| \leq 1$
(4) even polynomial $P \in \mathbb{R}[x]$ of degree d (even $\implies$ for all $x \in \mathbb{R}$, $P(x) = P(-x)$)

Output: estimate $\hat{z} \in \mathbb{C}$ s.t. $|\hat{z} - v^\dagger P(\sqrt{A^\dagger A})u| \leq \epsilon$

Lemma: Dequantizing SVT

$\forall \epsilon \in (0, 1]$ and $\zeta \leq \epsilon/8$, SVT solvable classically with probability $1 - 1/\operatorname{poly}(N)$ in $O^*((s^{2d+1})/\epsilon^2)$ time.

Proof sketch for dequantizing SVT

SVT(s, ϵ, ζ) (singular value transform)

Input: (1) query-access to s-sparse matrix $A \in \mathbb{C}^{M\times N}$ with $\|A\| \leq 1$
(2) query-access to $u \in \mathbb{C}^N$ s.t. $\|u\| \leq 1$
(3) ζ-samplable $v \in \mathbb{C}^N$ s.t. $\|v\| \leq 1$
(4) even polynomial $P \in \mathbb{R}[x]$ of degree d (recall: even $\implies$ for all $x \in \mathbb{R}$, $P(x) = P(-x)$)

Output: estimate $\hat{z} \in \mathbb{C}$ s.t. $|\hat{z} - v^\dagger P(\sqrt{A^\dagger A})u| \leq \epsilon$

Proof sketch.

Idea (à la [Tang 2019]): Compute r random entries of $\langle v, P(\sqrt{A^\dagger A})u\rangle$, take arithmetic mean:

Proof sketch for dequantizing SVT

SVT(s, ϵ, ζ) (singular value transform)

Input: (1) query-access to s-sparse matrix $A \in \mathbb{C}^{M\times N}$ with $\|A\| \leq 1$
(2) query-access to $u \in \mathbb{C}^N$ s.t. $\|u\| \leq 1$
(3) ζ-samplable $v \in \mathbb{C}^N$ s.t. $\|v\| \leq 1$
(4) even polynomial $P \in \mathbb{R}[x]$ of degree d (recall: even $\implies$ for all $x \in \mathbb{R}$, $P(x) = P(-x)$)

Output: estimate $\hat{z} \in \mathbb{C}$ s.t. $|\hat{z} - v^\dagger P(\sqrt{A^\dagger A})u| \leq \epsilon$

Proof sketch.

Idea (à la [Tang 2019]): Compute r random entries of $\langle v, P(\sqrt{A^\dagger A})u\rangle$, take arithmetic mean:

1. Set avg $= 0$
2. Repeat $r \in \Theta(1/\epsilon^2)$ times:

Proof sketch for dequantizing SVT

SVT(s, ϵ, ζ) (singular value transform)

Input: (1) query-access to s-sparse matrix $A \in \mathbb{C}^{M\times N}$ with $\|A\| \leq 1$
(2) query-access to $u \in \mathbb{C}^N$ s.t. $\|u\| \leq 1$
(3) ζ-samplable $v \in \mathbb{C}^N$ s.t. $\|v\| \leq 1$
(4) even polynomial $P \in \mathbb{R}[x]$ of degree d (recall: even $\implies$ for all $x \in \mathbb{R}$, $P(x) = P(-x)$)

Output: estimate $\hat{z} \in \mathbb{C}$ s.t. $|\hat{z} - v^\dagger P(\sqrt{A^\dagger A})u| \leq \epsilon$

Proof sketch.

Idea (à la [Tang 2019]): Compute r random entries of $\langle v, P(\sqrt{A^\dagger A})u\rangle$, take arithmetic mean:

1. Set avg $= 0$
2. Repeat $r \in \Theta(1/\epsilon^2)$ times:
   - Via ζ-sampling of v, sample index $j \in \{1, \ldots, N\}$ (i.e. w.p. $p(j) \approx |v_j|^2 / \|v\|^2$)
   - Via query access, compute entry v_j

Proof sketch for dequantizing SVT

SVT(s, ϵ, ζ) (singular value transform)

Input: (1) query-access to s-sparse matrix $A \in \mathbb{C}^{M \times N}$ with $\|A\| \leq 1$
(2) query-access to $u \in \mathbb{C}^N$ s.t. $\|u\| \leq 1$
(3) ζ-samplable $v \in \mathbb{C}^N$ s.t. $\|v\| \leq 1$
(4) even polynomial $P \in \mathbb{R}[x]$ of degree d (recall: even $\implies$ for all $x \in \mathbb{R}$, $P(x) = P(-x)$)

Output: estimate $\hat{z} \in \mathbb{C}$ s.t. $|\hat{z} - v^\dagger P(\sqrt{A^\dagger A})u| \leq \epsilon$

Proof sketch.

Idea (à la [Tang 2019]): Compute r random entries of $\langle v, P(\sqrt{A^\dagger A})u\rangle$, take arithmetic mean:

1. Set avg $= 0$
2. Repeat $r \in \Theta(1/\epsilon^2)$ times:
   - Via ζ-sampling of v, sample index $j \in \{1, \ldots, N\}$ (i.e. w.p. $p(j) \approx |v_j|^2 / \|v\|^2$)
   - Via query access, compute entry v_j
   - Via s-sparsity of A, compute entry j of $w := P(\sqrt{A^\dagger A})u$ (do this recursively)

Proof sketch for dequantizing SVT

SVT(s, ϵ, ζ) (singular value transform)

Input: (1) query-access to s-sparse matrix $A \in \mathbb{C}^{M\times N}$ with $\|A\| \leq 1$
(2) query-access to $u \in \mathbb{C}^N$ s.t. $\|u\| \leq 1$
(3) ζ-samplable $v \in \mathbb{C}^N$ s.t. $\|v\| \leq 1$
(4) even polynomial $P \in \mathbb{R}[x]$ of degree d (recall: even $\implies$ for all $x \in \mathbb{R}$, $P(x) = P(-x)$)

Output: estimate $\hat{z} \in \mathbb{C}$ s.t. $|\hat{z} - v^\dagger P(\sqrt{A^\dagger A})u| \leq \epsilon$

Proof sketch.

Idea (à la [Tang 2019]): Compute r random entries of $\langle v, P(\sqrt{A^\dagger A})u\rangle$, take arithmetic mean:

1. Set avg $= 0$
2. Repeat $r \in \Theta(1/\epsilon^2)$ times:
   - Via ζ-sampling of v, sample index $j \in \{1, \ldots, N\}$ (i.e. w.p. $p(j) \approx |v_j|^2 \,/\, \|v\|^2$)
   - Via query access, compute entry v_j
   - Via s-sparsity of A, compute entry j of $w := P(\sqrt{A^\dagger A})u$ (do this recursively)
   - Update avg $=$ avg $+ (w_j m^2)/(v_j r)$

Proof sketch for dequantizing SVT

SVT(s, ϵ, ζ) (singular value transform)

Input: (1) query-access to s-sparse matrix $A \in \mathbb{C}^{M \times N}$ with $\|A\| \leq 1$
(2) query-access to $u \in \mathbb{C}^N$ s.t. $\|u\| \leq 1$
(3) ζ-samplable $v \in \mathbb{C}^N$ s.t. $\|v\| \leq 1$
(4) even polynomial $P \in \mathbb{R}[x]$ of degree d (recall: even $\implies$ for all $x \in \mathbb{R}$, $P(x) = P(-x)$)

Output: estimate $\hat{z} \in \mathbb{C}$ s.t. $|\hat{z} - v^\dagger P(\sqrt{A^\dagger A})u| \leq \epsilon$

Proof sketch.

Idea (à la [Tang 2019]): Compute r random entries of $\langle v, P(\sqrt{A^\dagger A})u\rangle$, take arithmetic mean:

1. Set avg $= 0$
2. Repeat $r \in \Theta(1/\epsilon^2)$ times:
   - Via ζ-sampling of v, sample index $j \in \{1, \dots, N\}$ (i.e. w.p. $p(j) \approx |v_j|^2 / \|v\|^2$)
   - Via query access, compute entry v_j
   - Via s-sparsity of A, compute entry j of $w := P(\sqrt{A^\dagger A})u$ (do this recursively)
   - Update avg $=$ avg $+ (w_j m^2)/(v_j r)$

Correctness: High probability bound obtained via Chebyshev's inequality

Summary

- Overview of quantum algorithms over the decades
- Quantum algorithm for solving linear systems
- Quantum Singular Value Transform
- Dequantization - beware the power of state preparation

References

- Harrow, Hassadim, Lloyd. Quantum algorithm for solving linear systems of equations, 2009.
- Gilyén, Su, Low, Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics, 2019.
- Tang. A quantum-inspired classical algorithm for recommendation systems, 2019.
- Gharibian, Le Gall. Dequantizing the Quantum Singular Value Transformation: Hardness and Applications to Quantum Chemistry and the Quantum PCP Conjecture, 2022.

Course notes/videos for Intro to Quantum Computation and Quantum Complexity Theory:

- See `https://groups.uni-paderborn.de/fg-qi/teaching.html`.

References

- Harrow, Hassadim, Lloyd. Quantum algorithm for solving linear systems of equations, 2009.
- Gilyén, Su, Low, Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics, 2019.
- Tang. A quantum-inspired classical algorithm for recommendation systems, 2019.
- Gharibian, Le Gall. Dequantizing the Quantum Singular Value Transformation: Hardness and Applications to Quantum Chemistry and the Quantum PCP Conjecture, 2022.

Course notes/videos for Intro to Quantum Computation and Quantum Complexity Theory:

- See `https://groups.uni-paderborn.de/fg-qi/teaching.html`.