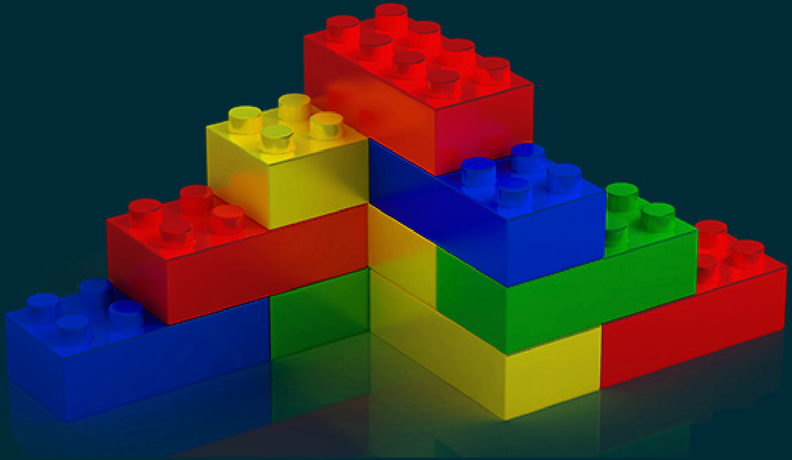
Quantum singular value transformation

András Gilyén

ELTE kvantum-infó szeminárium 2020 október 1

Quantum algorithm design



Many quantum algorithms have a common structure!

Szegedy quantum walk

Discrete-time Markov-chain on a weighted graph

Transition probability in one step (stochastic matrix)

$$P_{vu} = \Pr(\text{step to } v \,|\, \text{being at } u) = \frac{w_{vu}}{\sum_{v' \in U} w_{v'u}}$$

Szegedy quantum walk

Discrete-time Markov-chain on a weighted graph

Transition probability in one step (stochastic matrix)

$$P_{vu} = \Pr(\text{step to } v \mid \text{being at } u) = \frac{w_{vu}}{\sum_{v' \in U} w_{v'u}}$$

Szegedy walk operator

$$W' := U^\dagger \cdot \mathrm{SWAP} \cdot U$$

$$W := U^\dagger \cdot \mathrm{SWAP} \cdot U((2|0\rangle\langle 0| \otimes I) - I)$$

A block-encoding of the (symmetric) Markov chain: $(\langle 0| \otimes I) W' (|0\rangle \otimes I) = P$

Szegedy quantum walk

Discrete-time Markov-chain on a weighted graph

Transition probability in one step (stochastic matrix)

$$P_{vu} = \Pr(\text{step to } v \,|\, \text{being at } u) = \frac{w_{vu}}{\sum_{v' \in U} w_{v'u}}$$

Szegedy walk operator

$$W' := U^{\dagger} \cdot \mathrm{SWAP} \cdot U$$
$$W := U^{\dagger} \cdot \mathrm{SWAP} \cdot U((2|0\rangle\langle 0| \otimes I) - I)$$

A block-encoding of the (symmetric) Markov chain: $(\langle 0| \otimes I) W' (|0\rangle \otimes I) = P$

Multiple steps of the quantum walk: $(\langle 0| \otimes I) W^k (|0\rangle \otimes I) = T_k(P)$

$[T_k(x) = \cos(k \arccos(x))$ Chebyshev polynomials: $T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)]$

Szegedy quantum walk

Discrete-time Markov-chain on a weighted graph

Transition probability in one step (stochastic matrix)

$$P_{vu} = \Pr(\text{step to } v \,|\, \text{being at } u) = \frac{w_{vu}}{\sum_{v' \in U} w_{v'u}}$$

Szegedy walk operator

$$W' := U^\dagger \cdot \mathrm{SWAP} \cdot U$$
$$W := U^\dagger \cdot \mathrm{SWAP} \cdot U((2|0\rangle\langle 0| \otimes I) - I)$$

A block-encoding of the (symmetric) Markov chain: $(\langle 0| \otimes I) W' (|0\rangle \otimes I) = P$

Multiple steps of the quantum walk: $(\langle 0| \otimes I) W^k (|0\rangle \otimes I) = T_k(P)$

[$T_k(x) = \cos(k \arccos(x))$ Chebyshev polynomials: $T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$]

Proof: Proceed by induction, observe $T_0(P) = I$ ✓, $T_1(P) = P$ ✓

Szegedy quantum walk

Discrete-time Markov-chain on a weighted graph

Transition probability in one step (stochastic matrix)

$$P_{vu} = \Pr(\text{step to } v \,|\, \text{being at } u) = \frac{w_{vu}}{\sum_{v' \in U} w_{v'u}}$$

Szegedy walk operator

$$W' := U^\dagger \cdot \mathrm{SWAP} \cdot U$$

$$W := U^\dagger \cdot \mathrm{SWAP} \cdot U((2|0\rangle\langle 0| \otimes I) - I)$$

A block-encoding of the (symmetric) Markov chain: $(\langle 0| \otimes I) W' (|0\rangle \otimes I) = P$

Multiple steps of the quantum walk: $(\langle 0| \otimes I) W^k (|0\rangle \otimes I) = T_k(P)$

$[T_k(x) = \cos(k \arccos(x))$ Chebyshev polynomials: $T_{k+1}(x) = 2x T_k(x) - T_{k-1}(x)]$

Proof: Proceed by induction, observe $T_0(P) = I \checkmark$, $T_1(P) = P \checkmark$

$$(\langle 0| \otimes I) W^{k+1} (|0\rangle \otimes I) = (\langle 0| \otimes I) W' ((2|0\rangle\langle 0| \otimes I) - I) W^k (|0\rangle \otimes I) =$$

$$= \underbrace{(\langle 0| \otimes I) W' (2|0\rangle}_{2P} \underbrace{\langle 0| \otimes I) W^k (|0\rangle \otimes I)}_{T_k(P)} - \underbrace{(\langle 0| \otimes I) W^{k-1} (|0\rangle \otimes I)}_{T_{k-1}(P)}$$

Szegedy quantum walk

Discrete-time Markov-chain on a weighted graph

Transition probability in one step (stochastic matrix)

$$P_{vu} = \Pr(\text{step to } v \,|\, \text{being at } u) = \frac{w_{vu}}{\sum_{v' \in U} w_{v'u}}$$

Szegedy walk operator

$$W' := U^\dagger \cdot \mathrm{SWAP} \cdot U$$

$$W := U^\dagger \cdot \mathrm{SWAP} \cdot U((2|0\rangle\langle 0| \otimes I) - I)$$

A block-encoding of the (symmetric) Markov chain: $(\langle 0| \otimes I) W' (|0\rangle \otimes I) = P$

Multiple steps of the quantum walk: $(\langle 0| \otimes I) W^k (|0\rangle \otimes I) = T_k(P)$

[$T_k(x) = \cos(k \arccos(x))$ Chebyshev polynomials: $T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$]

Proof: Proceed by induction, observe $T_0(P) = I \checkmark$, $T_1(P) = P \checkmark$

$$(\langle 0| \otimes I) W^{k+1} (|0\rangle \otimes I) = (\langle 0| \otimes I) W' ((2|0\rangle\langle 0| \otimes I) - I) W^k (|0\rangle \otimes I) =$$

$$= \underbrace{(\langle 0| \otimes I) W' (2|0\rangle}_{2P} \underbrace{\langle 0| \otimes I) W^k (|0\rangle \otimes I)}_{T_k(P)} - \underbrace{(\langle 0| \otimes I) W^{k-1} (|0\rangle \otimes I)}_{T_{k-1}(P)}$$

Grover search and amplitude amplification

Amplitude amplification problem

Given U such that

$$U\left|\bar{0}\right\rangle = \sqrt{p}|0\rangle\left|\psi_{\text{good}}\right\rangle + \sqrt{1-p}|1\rangle|\psi_{\text{bad}}\rangle,$$

Grover search and amplitude amplification

Amplitude amplification problem

Given U such that

$$U\left|\bar{0}\right\rangle = \sqrt{p}|0\rangle\left|\psi_{\text{good}}\right\rangle + \sqrt{1-p}|1\rangle|\psi_{\text{bad}}\rangle,$$

prepare $\left|\psi_{\text{good}}\right\rangle$ (with high probability).

Algorithm and its success probability

$$U \cdots [2|\bar{0}\rangle\langle\bar{0}| - I]\, U^{\dagger}\, [(|0\rangle\langle 0| - |1\rangle\langle 1|) \otimes I]\, U[2|\bar{0}\rangle\langle\bar{0}| - I]\, U^{\dagger}\, [(|0\rangle\langle 0| - |1\rangle\langle 1|) \otimes I]\, U$$

Grover search and amplitude amplification

Amplitude amplification problem

Given U such that

$$U\left|\bar{0}\right\rangle = \sqrt{p}|0\rangle\left|\psi_{\text{good}}\right\rangle + \sqrt{1-p}|1\rangle|\psi_{\text{bad}}\rangle,$$

prepare $\left|\psi_{\text{good}}\right\rangle$ (with high probability).

Algorithm and its success probability

$$U \cdots [2|\bar{0}\rangle\langle\bar{0}| - I]\, U^{\dagger}\, [(|0\rangle\langle 0| - |1\rangle\langle 1|) \otimes I]\, U[2|\bar{0}\rangle\langle\bar{0}| - I]\, U^{\dagger}\, [(|0\rangle\langle 0| - |1\rangle\langle 1|) \otimes I]\, U$$

amplitude of $\left|\psi_{\text{good}}\right\rangle$ after k iterations:

$$\pm\sin((2k+1)\alpha), \qquad \text{where } \alpha = \arcsin(\sqrt{p})$$

Grover search and amplitude amplification

Amplitude amplification problem

Given U such that

$$U|\bar{0}\rangle = \sqrt{p}|0\rangle|\psi_{\text{good}}\rangle + \sqrt{1-p}|1\rangle|\psi_{\text{bad}}\rangle,$$

prepare $|\psi_{\text{good}}\rangle$ (with high probability).

Algorithm and its success probability

$$U \cdots [2|\bar{0}\rangle\langle\bar{0}| - I]\, U^\dagger\, [(|0\rangle\langle 0| - |1\rangle\langle 1|) \otimes I]\, U[2|\bar{0}\rangle\langle\bar{0}| - I]\, U^\dagger\, [(|0\rangle\langle 0| - |1\rangle\langle 1|) \otimes I]\, U$$

amplitude of $|\psi_{\text{good}}\rangle$ after k iterations:

$$\pm \sin((2k+1)\alpha), \quad \text{where } \alpha = \arcsin(\sqrt{p})$$

which is

$$\pm \cos((2k+1)\arccos(\sqrt{p})) = \pm T_{2k+1}(\sqrt{p}).$$

Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map.

Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map. Suppose that

$$U = \begin{bmatrix} A & . \\ . & . \end{bmatrix} = \begin{bmatrix} \sum_i \varsigma_i |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix}$$

Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map. Suppose that

$$U = \begin{bmatrix} A & . \\ . & . \end{bmatrix} = \begin{bmatrix} \sum_i \varsigma_i |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix} \Longrightarrow U_\Phi = \begin{bmatrix} \sum_i P(\varsigma_i) |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix},$$

Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map. Suppose that

$$U = \begin{bmatrix} A & . \\ . & . \end{bmatrix} = \begin{bmatrix} \sum_i \varsigma_i |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix} \Longrightarrow U_\Phi = \begin{bmatrix} \sum_i P(\varsigma_i) |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix},$$

where $\Phi(P) \in \mathbb{R}^d$ is efficiently computable and U_Φ is the following circuit:

Quantum Singular Value Transformation (QSVT)

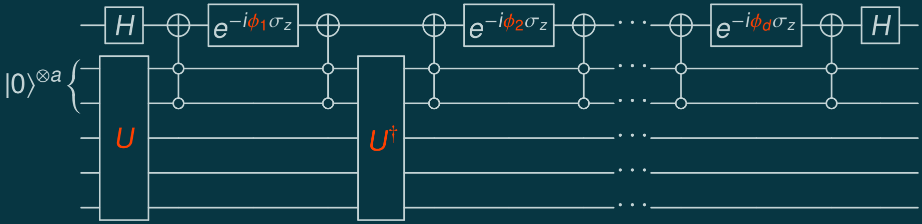
Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map. Suppose that

$$U = \begin{bmatrix} A & . \\ . & . \end{bmatrix} = \begin{bmatrix} \sum_i \varsigma_i |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix} \implies U_\Phi = \begin{bmatrix} \sum_i P(\varsigma_i) |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix},$$

where $\Phi(P) \in \mathbb{R}^d$ is efficiently computable and U_Φ is the following circuit:

Alternating phase modulation sequence $U_\Phi :=$



Quantum Singular Value Transformation (QSVT)

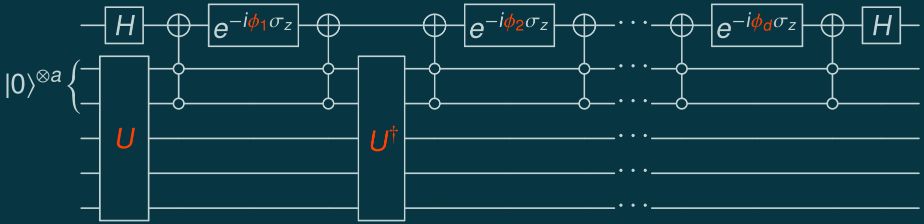
Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map. Suppose that

$$U = \begin{bmatrix} A & . \\ . & . \end{bmatrix} = \begin{bmatrix} \sum_i \varsigma_i |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix} \Longrightarrow U_\Phi = \begin{bmatrix} \sum_i P(\varsigma_i) |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix},$$

where $\Phi(P) \in \mathbb{R}^d$ is efficiently computable and U_Φ is the following circuit:

Alternating phase modulation sequence $U_\Phi :=$



Simmilar result holds for even polynomials.

Quantum algorithm design

Outline

Motivating example - the HHL algorithm

We want to solve large systems of linear equations

$$Ax = b.$$

A quantum computer can nicely work with exponential sized matrices!
Given $|b\rangle$, we can prepare a solution $\propto A^{-1}|b\rangle$.

Outline

Motivating example - the HHL algorithm

We want to solve large systems of linear equations

$$Ax = b.$$

A quantum computer can nicely work with exponential sized matrices!
Given $|b\rangle$, we can prepare a solution $\propto A^{-1}|b\rangle$.

Matrix arithmetic on a quantum computer using block-encoding

$$\text{Target}: \; A; \; \text{Implementation}: \; U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix}; \; \text{Algorithm}: \; U' = \begin{bmatrix} f(A) & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

In HHL $f(x) = \frac{1}{x}$. Use Singular Value Transformation to implement it!

Outline

Motivating example - the HHL algorithm

We want to solve large systems of linear equations

$$Ax = b.$$

A quantum computer can nicely work with exponential sized matrices!
Given $|b\rangle$, we can prepare a solution $\propto A^{-1}|b\rangle$.

Matrix arithmetic on a quantum computer using block-encoding

$$\text{Target}: A; \text{ Implementation}: U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix}; \text{ Algorithm}: U' = \begin{bmatrix} f(A) & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

In HHL $f(x) = \frac{1}{x}$. Use Singular Value Transformation to implement it!

Applications

- *Optimal Hamiltonian simulation [Low et al.], Quantum walks [Szegedy]*
- *Fixed point [Yoder et al.] and Oblivious ampl. ampl. [Berry et al.]*
- *HHL, Regression [Chakraborty et al.], ML [Kerendis & Prakash], Property testing, . . .*

Block-encoding

A way to represent large matrices on a quantum computer efficiently

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \Longleftrightarrow \quad A = \left(\langle 0|^{a} \otimes I\right) U \left(|0\rangle^{b} \otimes I\right).$$

Block-encoding

A way to represent large matrices on a quantum computer efficiently

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \Longleftrightarrow \quad A = (\langle 0|^a \otimes I)\, U \big(|0\rangle^b \otimes I\big).$$

One can efficiently construct block-encodings of

Block-encoding

A way to represent large matrices on a quantum computer efficiently

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix} \iff A = (\langle 0|^a \otimes I)\, U \left(|0\rangle^b \otimes I\right).$$

One can efficiently construct block-encodings of

- an efficiently implementable unitary U,

Block-encoding

A way to represent large matrices on a quantum computer efficiently

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \Longleftrightarrow \quad A = (\langle 0|^{a} \otimes I)\, U \left(|0\rangle^{b} \otimes I\right).$$

One can efficiently construct block-encodings of

- an efficiently implementable unitary U,
- a sparse matrix with efficiently computable elements,

Block-encoding

A way to represent large matrices on a quantum computer efficiently

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \Longleftrightarrow \quad A = (\langle 0|^a \otimes I)\, U \left(|0\rangle^b \otimes I\right).$$

One can efficiently construct block-encodings of

- an efficiently implementable unitary U,
- a sparse matrix with efficiently computable elements,
- a matrix stored in a clever data-structure in a QRAM,

Block-encoding

A way to represent large matrices on a quantum computer efficiently

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \Longleftrightarrow \quad A = (\langle 0|^a \otimes I)\, U \left(|0\rangle^b \otimes I\right).$$

One can efficiently construct block-encodings of

- an efficiently implementable unitary U,
- a sparse matrix with efficiently computable elements,
- a matrix stored in a clever data-structure in a QRAM,
- a density operator ρ given a unitary preparing its purification.

Block-encoding

A way to represent large matrices on a quantum computer efficiently

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \Longleftrightarrow \quad A = (\langle 0|^a \otimes I)\, U \big(|0\rangle^b \otimes I\big).$$

One can efficiently construct block-encodings of

- an efficiently implementable unitary U,
- a sparse matrix with efficiently computable elements,
- a matrix stored in a clever data-structure in a QRAM,
- a density operator ρ given a unitary preparing its purification.
- a POVM operator M given we can sample from the rand.var.: $\mathrm{Tr}(\rho M)$,

Example: Block-encoding sparse matrices

Suppose that A is s-sparse and $|A_{ij}| \leq 1$ for all i, j indices.

Example: Block-encoding sparse matrices

Suppose that A is s-sparse and $|A_{ij}| \leq 1$ for all i, j indices. Given "sparse-access" we can efficiently implement unitaries preparing "rows"

Example: Block-encoding sparse matrices

Suppose that A is s-sparse and $|A_{ij}| \leq 1$ for all i, j indices. Given "sparse-access" we can efficiently implement unitaries preparing "rows"

$$R: |0\rangle|0\rangle|i\rangle \to |0\rangle \sum_k \frac{(\sqrt{A_{ik}})^*}{\sqrt{s}}|i\rangle|k\rangle + |1\rangle|i\rangle|\text{garbage}\rangle,$$

Example: Block-encoding sparse matrices

Suppose that A is s-sparse and $|A_{ij}| \leq 1$ for all i, j indices. Given ”sparse-access “ we can efficiently implement unitaries preparing ”rows“

$$R : |0\rangle|0\rangle|i\rangle \rightarrow |0\rangle \sum_k \frac{(\sqrt{A_{ik}})^*}{\sqrt{s}}|i\rangle|k\rangle + |1\rangle|i\rangle|\text{garbage}\rangle,$$

and ”columns“

$$C : |0\rangle|0\rangle|j\rangle \rightarrow |0\rangle \sum_\ell \frac{\sqrt{A_{\ell j}}}{\sqrt{s}}|\ell\rangle|j\rangle + |2\rangle|j\rangle|\text{garbage}\rangle,$$

Example: Block-encoding sparse matrices

Suppose that A is s-sparse and $|A_{ij}| \leq 1$ for all i, j indices. Given "sparse-access" we can efficiently implement unitaries preparing "rows"

$$R\colon |0\rangle|0\rangle|i\rangle \to |0\rangle \sum_k \frac{(\sqrt{A_{ik}})^*}{\sqrt{s}}|i\rangle|k\rangle + |1\rangle|i\rangle|\text{garbage}\rangle,$$

and "columns"

$$C\colon |0\rangle|0\rangle|j\rangle \to |0\rangle \sum_\ell \frac{\sqrt{A_{\ell j}}}{\sqrt{s}}|\ell\rangle|j\rangle + |2\rangle|j\rangle|\text{garbage}\rangle,$$

They form a block-encoding of A/s:

$$\langle 0|\langle 0|\langle i|R^\dagger C|0\rangle|0\rangle|j\rangle$$

Example: Block-encoding sparse matrices

Suppose that A is s-sparse and $|A_{ij}| \leq 1$ for all i, j indices. Given "sparse-access" we can efficiently implement unitaries preparing "rows"

$$R: |0\rangle|0\rangle|i\rangle \to |0\rangle \sum_k \frac{(\sqrt{A_{ik}})^*}{\sqrt{s}}|i\rangle|k\rangle + |1\rangle|i\rangle|\text{garbage}\rangle,$$

and "columns"

$$C: |0\rangle|0\rangle|j\rangle \to |0\rangle \sum_\ell \frac{\sqrt{A_{\ell j}}}{\sqrt{s}}|\ell\rangle|j\rangle + |2\rangle|j\rangle|\text{garbage}\rangle,$$

They form a block-encoding of A/s:

$$\langle 0|\langle 0|\langle i|R^\dagger C|0\rangle|0\rangle|j\rangle = (R|0\rangle|0\rangle|i\rangle)^\dagger \cdot (C|0\rangle|0\rangle|j\rangle)$$

Example: Block-encoding sparse matrices

Suppose that A is s-sparse and $|A_{ij}| \leq 1$ for all i, j indices. Given ”sparse-access “ we can efficiently implement unitaries preparing ”rows“

$$R: |0\rangle|0\rangle|i\rangle \to |0\rangle \sum_k \frac{(\sqrt{A_{ik}})^*}{\sqrt{s}}|i\rangle|k\rangle + |1\rangle|i\rangle|\text{garbage}\rangle,$$

and ”columns“

$$C: |0\rangle|0\rangle|j\rangle \to |0\rangle \sum_\ell \frac{\sqrt{A_{\ell j}}}{\sqrt{s}}|\ell\rangle|j\rangle + |2\rangle|j\rangle|\text{garbage}\rangle,$$

They form a block-encoding of A/s:

$$\langle 0|\langle 0|\langle i|R^\dagger C|0\rangle|0\rangle|j\rangle = (R|0\rangle|0\rangle|i\rangle)^\dagger \cdot (C|0\rangle|0\rangle|j\rangle) = \left(\sum_k \frac{(\sqrt{A_{ik}})^*}{\sqrt{s}}|i\rangle|k\rangle\right)^\dagger \left(\sum_\ell \frac{\sqrt{A_{\ell j}}}{\sqrt{s}}|\ell\rangle|j\rangle\right)$$

Example: Block-encoding sparse matrices

Suppose that A is s-sparse and $|A_{ij}| \leq 1$ for all i, j indices. Given "sparse-access " we can efficiently implement unitaries preparing "rows"

$$R\colon |0\rangle|0\rangle|i\rangle \to |0\rangle \sum_k \frac{(\sqrt{A_{ik}})^*}{\sqrt{s}}|i\rangle|k\rangle + |1\rangle|i\rangle|\text{garbage}\rangle,$$

and "columns"

$$C\colon |0\rangle|0\rangle|j\rangle \to |0\rangle \sum_\ell \frac{\sqrt{A_{\ell j}}}{\sqrt{s}}|\ell\rangle|j\rangle + |2\rangle|j\rangle|\text{garbage}\rangle,$$

They form a block-encoding of A/s:

$$\langle 0|\langle 0|\langle i|R^\dagger C|0\rangle|0\rangle|j\rangle = (R|0\rangle|0\rangle|i\rangle)^\dagger \cdot (C|0\rangle|0\rangle|j\rangle) = \left(\sum_k \frac{(\sqrt{A_{ik}})^*}{\sqrt{s}}|i\rangle|k\rangle\right)^\dagger \left(\sum_\ell \frac{\sqrt{A_{\ell j}}}{\sqrt{s}}|\ell\rangle|j\rangle\right) = \frac{A_{ij}}{s}$$

Efficient matrix arithmetics

Efficient matrix arithmetics

Implementing arithmetic operations on block-encoded matrices

- Given block-encodings A_j we can implement convex combinations.

Efficient matrix arithmetics

Implementing arithmetic operations on block-encoded matrices

- Given block-encodings A_j we can implement convex combinations.
- Given block-encodings A, B we can implement block-encoding of AB.

Efficient matrix arithmetics

Implementing arithmetic operations on block-encoded matrices

- Given block-encodings A_j we can implement convex combinations.
- Given block-encodings A, B we can implement block-encoding of AB.

Linear combination of (non-)unitary matrices [Childs and Wiebe '12, Berry et al. '15]

Suppose that $U = \sum_i |i\rangle\langle i| \otimes U_i$, and $P : |0\rangle \mapsto \sum_i \sqrt{p_i}|i\rangle$ for $p_i \in [0, 1]$.

Efficient matrix arithmetics

Implementing arithmetic operations on block-encoded matrices

- Given block-encodings A_j we can implement convex combinations.
- Given block-encodings A, B we can implement block-encoding of AB.

Linear combination of (non-)unitary matrices [Childs and Wiebe '12, Berry et al. '15]

Suppose that $U = \sum_i |i\rangle\langle i| \otimes U_i$, and $P : |0\rangle \mapsto \sum_i \sqrt{p_i}|i\rangle$ for $p_i \in [0, 1]$.
Then $(P^\dagger \otimes I)U(P \otimes I)$ is a block-encoding of $\sum_i p_i U_i$.

Efficient matrix arithmetics

Implementing arithmetic operations on block-encoded matrices

- Given block-encodings A_j we can implement convex combinations.
- Given block-encodings A, B we can implement block-encoding of AB.

Linear combination of (non-)unitary matrices [Childs and Wiebe '12, Berry et al. '15]

Suppose that $U = \sum_i |i\rangle\langle i| \otimes U_i$, and $P : |0\rangle \mapsto \sum_i \sqrt{p_i}|i\rangle$ for $p_i \in [0, 1]$.
Then $(P^\dagger \otimes I)U(P \otimes I)$ is a block-encoding of $\sum_i p_i U_i$.
In particular if $(\langle 0| \otimes I)U_i(|0\rangle \otimes I) = A_i$, then it is a block-encoding of

$$\sum_i p_i A_i.$$

Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map.

Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map. Suppose that

$$U = \begin{bmatrix} A & . \\ . & . \end{bmatrix} = \begin{bmatrix} \sum_i \varsigma_i |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix}$$

Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map. Suppose that

$$U = \begin{bmatrix} A & . \\ . & . \end{bmatrix} = \begin{bmatrix} \sum_i \varsigma_i |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix} \Longrightarrow U_\Phi = \begin{bmatrix} \sum_i P(\varsigma_i) |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix},$$

Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map. Suppose that

$$U = \begin{bmatrix} A & . \\ . & . \end{bmatrix} = \begin{bmatrix} \sum_i \varsigma_i |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix} \implies U_\Phi = \begin{bmatrix} \sum_i P(\varsigma_i) |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix},$$

where $\Phi(P) \in \mathbb{R}^d$ is efficiently computable and U_Φ is the following circuit:

Quantum Singular Value Transformation (QSVT)

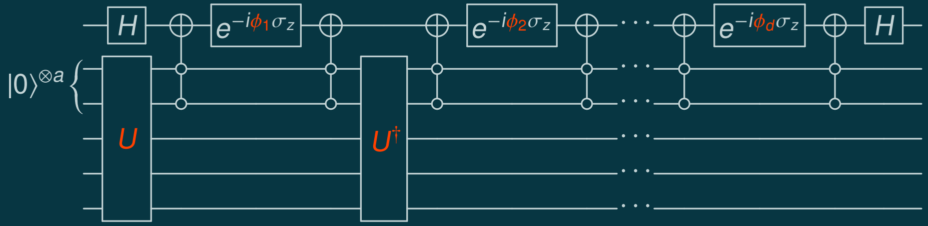
Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map. Suppose that

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix} = \begin{bmatrix} \sum_i \varsigma_i |w_i\rangle\langle v_i| & \cdot \\ \cdot & \cdot \end{bmatrix} \implies U_\Phi = \begin{bmatrix} \sum_i P(\varsigma_i) |w_i\rangle\langle v_i| & \cdot \\ \cdot & \cdot \end{bmatrix},$$

where $\Phi(P) \in \mathbb{R}^d$ is efficiently computable and U_Φ is the following circuit:

Alternating phase modulation sequence $U_\Phi :=$



Quantum Singular Value Transformation (QSVT)

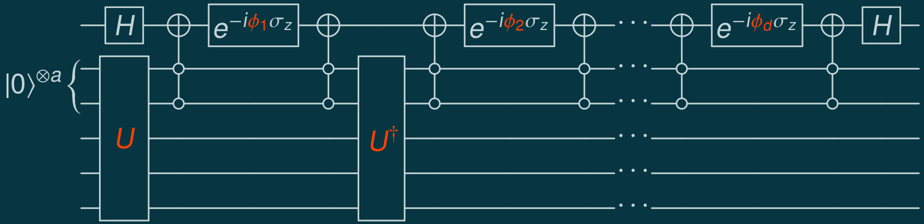
Our main theorem about QSVT

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d odd polynomial map. Suppose that

$$U = \begin{bmatrix} A & . \\ . & . \end{bmatrix} = \begin{bmatrix} \sum_i \varsigma_i |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix} \Longrightarrow U_\Phi = \begin{bmatrix} \sum_i P(\varsigma_i) |w_i\rangle\langle v_i| & . \\ . & . \end{bmatrix},$$

where $\Phi(P) \in \mathbb{R}^d$ is efficiently computable and U_Φ is the following circuit:

Alternating phase modulation sequence $U_\Phi :=$



Simmilar result holds for even polynomials.

Quantum walks and Hermitian matrices

Connection to Szegedy quantum walks

Szegedy defined (2004) quantisation of a symmetric Markov chain M via a product of two reflection operators.

Quantum walks and Hermitian matrices

Connection to Szegedy quantum walks

Szegedy defined (2004) quantisation of a symmetric Markov chain M via a product of two reflection operators. We can understand his algorithm as

$$\text{Markov chain}: \; M; \;\; \text{Updates}: \; W' = \begin{bmatrix} M & \cdot \\ \cdot & \cdot \end{bmatrix}; \;\; \text{Walk}: \; W^n = \begin{bmatrix} T_n(M) & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

(T_d is the d-th Chebyshev polynomial of the first kind.)

Quantum walks and Hermitian matrices

Connection to Szegedy quantum walks

Szegedy defined (2004) quantisation of a symmetric Markov chain M via a product of two reflection operators. We can understand his algorithm as

$$\text{Markov chain}: \; M; \;\; \text{Updates}: \; W' = \begin{bmatrix} M & \cdot \\ \cdot & \cdot \end{bmatrix}; \;\; \text{Walk}: \; W^n = \begin{bmatrix} T_n(M) & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

(T_d is the d-th Chebyshev polynomial of the first kind.)
If we choose $\phi_j = \frac{\pi}{2}$ for all $j \in \{1, \ldots, d\}$, we get $P = \pm T_d$ in QSVT.

Quantum walks and Hermitian matrices

Connection to Szegedy quantum walks

Szegedy defined (2004) quantisation of a symmetric Markov chain M via a product of two reflection operators. We can understand his algorithm as

$$\text{Markov chain}: \; M; \;\; \text{Updates}: \; W' = \begin{bmatrix} M & \cdot \\ \cdot & \cdot \end{bmatrix}; \;\; \text{Walk}: \; W^n = \begin{bmatrix} T_n(M) & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

(T_d is the d-th Chebyshev polynomial of the first kind.)
If we choose $\phi_j = \frac{\pi}{2}$ for all $j \in \{1, \ldots, d\}$, we get $P = \pm T_d$ in QSVT.

Quantum Fast-Forwarding Markov Chains [Apers & Sarlette (2018)]

Quantum walks and Hermitian matrices

Connection to Szegedy quantum walks

Szegedy defined (2004) quantisation of a symmetric Markov chain M via a product of two reflection operators. We can understand his algorithm as

$$\text{Markov chain}: \ M; \ \text{Updates}: \ W' = \begin{bmatrix} M & \cdot \\ \cdot & \cdot \end{bmatrix}; \ \text{Walk}: \ W^n = \begin{bmatrix} T_n(M) & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

(T_d is the d-th Chebyshev polynomial of the first kind.)
If we choose $\phi_j = \frac{\pi}{2}$ for all $j \in \{1, \ldots, d\}$, we get $P = \pm T_d$ in QSVT.

Quantum Fast-Forwarding Markov Chains [Apers & Sarlette (2018)]

Simulate t classical steps using $\propto \sqrt{t}$ quantum operations.

Quantum walks and Hermitian matrices

Connection to Szegedy quantum walks

Szegedy defined (2004) quantisation of a symmetric Markov chain M via a product of two reflection operators. We can understand his algorithm as

$$\text{Markov chain}: \ M; \ \text{Updates}: \ W' = \begin{bmatrix} M & \cdot \\ \cdot & \cdot \end{bmatrix}; \ \text{Walk}: \ W^n = \begin{bmatrix} T_n(M) & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

(T_d is the d-th Chebyshev polynomial of the first kind.)
If we choose $\phi_j = \frac{\pi}{2}$ for all $j \in \{1, \ldots, d\}$, we get $P = \pm T_d$ in QSVT.

Quantum Fast-Forwarding Markov Chains [Apers & Sarlette (2018)]

Simulate t classical steps using $\propto \sqrt{t}$ quantum operations. I.e., implement

$$U' = \begin{bmatrix} M^t & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

Quantum walks and Hermitian matrices

Connection to Szegedy quantum walks

Szegedy defined (2004) quantisation of a symmetric Markov chain M via a product of two reflection operators. We can understand his algorithm as

$$\text{Markov chain}: \; M; \;\; \text{Updates}: \; W' = \begin{bmatrix} M & \cdot \\ \cdot & \cdot \end{bmatrix}; \;\; \text{Walk}: \; W^n = \begin{bmatrix} T_n(M) & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

(T_d is the d-th Chebyshev polynomial of the first kind.)
If we choose $\phi_j = \frac{\pi}{2}$ for all $j \in \{1, \ldots, d\}$, we get $P = \pm T_d$ in QSVT.

Quantum Fast-Forwarding Markov Chains [Apers & Sarlette (2018)]

Simulate t classical steps using $\propto \sqrt{t}$ quantum operations. I.e., implement

$$U' = \begin{bmatrix} M^t & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

Proof: x^t can be ε-apx. on $[-1, 1]$ with a degree-$\sqrt{2t \ln(2/\varepsilon)}$ polynomial.

The special case of Hermitian matrices

Singular value transf. = eigenvalue transf. [Low & Chuang (2017)]

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d even/odd polynomial map.

The special case of Hermitian matrices

Singular value transf. = eigenvalue transf. [Low & Chuang (2017)]

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d even/odd polynomial map.
If H is Hermitian, then $P(H)$ coincides with the singular value transform.

The special case of Hermitian matrices

Singular value transf. = eigenvalue transf. [Low & Chuang (2017)]

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d even/odd polynomial map.
If H is Hermitian, then $P(H)$ coincides with the singular value transform.

Removing parity constraint for Hermitian matrices

Let $P\colon [-1,1] \to [-\frac{1}{2}, \frac{1}{2}]$ be a degree-d polynomial map. Suppose that U is an a-qubit block-encoding of a Hermitian matrix H.

The special case of Hermitian matrices

Singular value transf. = eigenvalue transf. [Low & Chuang (2017)]

Let $P\colon [-1,1] \to [-1,1]$ be a degree-d even/odd polynomial map.
If H is Hermitian, then $P(H)$ coincides with the singular value transform.

Removing parity constraint for Hermitian matrices

Let $P\colon [-1,1] \to [-\frac{1}{2},\frac{1}{2}]$ be a degree-d polynomial map. Suppose that U is an a-qubit block-encoding of a Hermitian matrix H. We can implement

$$U' = \begin{bmatrix} P(H) & . \\ . & . \end{bmatrix},$$

using d times U and $U^\dagger$, 1 controlled U, and $\mathcal{O}(ad)$ extra two-qubit gates.

Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using σ_z phases?

Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using σ_z phases?

$$R(x) := \begin{bmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{bmatrix}; \quad e^{i\phi_0\sigma_z}R(x)e^{i\phi_1\sigma_z}\cdot\ldots\cdot R(x)e^{i\phi_d\sigma_z} = (*)?$$

Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using σ_z phases?

$$R(x) := \begin{bmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{bmatrix}; \quad e^{i\phi_0\sigma_z} R(x) e^{i\phi_1\sigma_z} \cdot \ldots \cdot R(x) e^{i\phi_d\sigma_z} = (*)?$$

Theorem: Basic characterization [Low, Yoder, Chuang (2016)]

Let $d \in \mathbb{N}$; for all $\Phi \in \mathbb{R}^{d+1}$ we have

$$(*) = \begin{bmatrix} P_{\mathbb{C}}(x) & iQ_{\mathbb{C}}(x)\sqrt{1-x^2} \\ iQ^*_{\mathbb{C}}(x)\sqrt{1-x^2} & P^*_{\mathbb{C}}(x) \end{bmatrix},$$

where $P_{\mathbb{C}}, Q_{\mathbb{C}} \in \mathbb{C}[x]$ are such that

Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using σ_z phases?

$$R(x) := \begin{bmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{bmatrix}; \quad e^{i\phi_0\sigma_z}R(x)e^{i\phi_1\sigma_z}\cdot\ldots\cdot R(x)e^{i\phi_d\sigma_z} = (*)?$$

Theorem: Basic characterization [Low, Yoder, Chuang (2016)]

Let $d \in \mathbb{N}$; for all $\Phi \in \mathbb{R}^{d+1}$ we have

$$(*) = \begin{bmatrix} P_{\mathbb{C}}(x) & iQ_{\mathbb{C}}(x)\sqrt{1-x^2} \\ iQ^*_{\mathbb{C}}(x)\sqrt{1-x^2} & P^*_{\mathbb{C}}(x) \end{bmatrix},$$

where $P_{\mathbb{C}}, Q_{\mathbb{C}} \in \mathbb{C}[x]$ are such that

(i) $\deg(P_{\mathbb{C}}) \leq d$ and $\deg(Q_{\mathbb{C}}) \leq d-1$, and

Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using σ_z phases?

$$R(x) := \begin{bmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{bmatrix}; \quad e^{i\phi_0\sigma_z} R(x) e^{i\phi_1\sigma_z} \cdot \ldots \cdot R(x) e^{i\phi_d\sigma_z} = (*)?$$

Theorem: Basic characterization [Low, Yoder, Chuang (2016)]

Let $d \in \mathbb{N}$; for all $\Phi \in \mathbb{R}^{d+1}$ we have

$$(*) = \begin{bmatrix} P_{\mathbb{C}}(x) & iQ_{\mathbb{C}}(x)\sqrt{1-x^2} \\ iQ^*_{\mathbb{C}}(x)\sqrt{1-x^2} & P^*_{\mathbb{C}}(x) \end{bmatrix},$$

where $P_{\mathbb{C}}, Q_{\mathbb{C}} \in \mathbb{C}[x]$ are such that

(i) $\deg(P_{\mathbb{C}}) \leq d$ and $\deg(Q_{\mathbb{C}}) \leq d-1$, and

(ii) $P_{\mathbb{C}}$ has parity-($d \mod 2$) and $Q_{\mathbb{C}}$ has parity-($d-1 \mod 2$), and

Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using σ_z phases?

$$R(x) := \begin{bmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{bmatrix}; \quad e^{i\phi_0\sigma_z}R(x)e^{i\phi_1\sigma_z}\cdot\ldots\cdot R(x)e^{i\phi_d\sigma_z} = (*)?$$

Theorem: Basic characterization [Low, Yoder, Chuang (2016)]

Let $d \in \mathbb{N}$; for all $\Phi \in \mathbb{R}^{d+1}$ we have

$$(*) = \begin{bmatrix} P_{\mathbb{C}}(x) & iQ_{\mathbb{C}}(x)\sqrt{1-x^2} \\ iQ^*_{\mathbb{C}}(x)\sqrt{1-x^2} & P^*_{\mathbb{C}}(x) \end{bmatrix},$$

where $P_{\mathbb{C}}, Q_{\mathbb{C}} \in \mathbb{C}[x]$ are such that

(i) $\deg(P_{\mathbb{C}}) \leq d$ and $\deg(Q_{\mathbb{C}}) \leq d-1$, and

(ii) $P_{\mathbb{C}}$ has parity-($d \mod 2$) and $Q_{\mathbb{C}}$ has parity-($d-1 \mod 2$), and

(iii) $\forall x \in [-1,1]$: $|P_{\mathbb{C}}(x)|^2 + (1-x^2)|Q_{\mathbb{C}}(x)|^2 = 1$.

Real quantum signal processing

Theorem: Focusing on the real part [Low, Yoder, Chuang (2016)]

Let $d \in \mathbb{N}$, and $P \in \mathbb{R}[x]$ be of degree d. There exists $\Phi \in \mathbb{R}^d$ such that

$$\prod_{j=1}^{d} \left(R(x) e^{i\phi_j \sigma_z} \right) = \begin{bmatrix} P_{\mathbb{C}}(x) & \cdot \\ \cdot & \cdot \end{bmatrix},$$

where $\Re[P_{\mathbb{C}}] = P$ if and only if

Real quantum signal processing

Theorem: Focusing on the real part [Low, Yoder, Chuang (2016)]

Let $d \in \mathbb{N}$, and $P \in \mathbb{R}[x]$ be of degree d. There exists $\Phi \in \mathbb{R}^d$ such that

$$\prod_{j=1}^{d} \left(R(x) e^{i\phi_j \sigma_z} \right) = \begin{bmatrix} P_{\mathbb{C}}(x) & \cdot \\ \cdot & \cdot \end{bmatrix},$$

where $\Re[P_{\mathbb{C}}] = P$ if and only if

(i) P has parity-($d \mod 2$), and

Real quantum signal processing

Theorem: Focusing on the real part [Low, Yoder, Chuang (2016)]

Let $d \in \mathbb{N}$, and $P \in \mathbb{R}[x]$ be of degree d. There exists $\Phi \in \mathbb{R}^d$ such that

$$\prod_{j=1}^{d} \left(R(x) e^{i\phi_j \sigma_z} \right) = \begin{bmatrix} P_{\mathbb{C}}(x) & \cdot \\ \cdot & \cdot \end{bmatrix},$$

where $\Re[P_{\mathbb{C}}] = P$ if and only if

(i) P has parity-($d \mod 2$), and

(ii) for all $x \in [-1, 1]$: $|P(x)| \leq 1$.

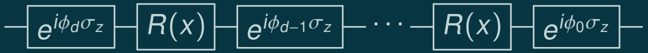
Implementing the real part of a polynomial map

Direct implementation

$$-\boxed{e^{i\phi_d\sigma_z}}-\boxed{R(x)}-\boxed{e^{i\phi_{d-1}\sigma_z}}-\cdots-\boxed{R(x)}-\boxed{e^{i\phi_0\sigma_z}}- = \begin{bmatrix} P_{\mathbb{C}}(x) & \cdot \\ \cdot & \cdot \end{bmatrix}$$

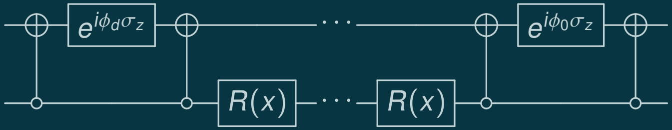
Implementing the real part of a polynomial map

Direct implementation



$$= \begin{bmatrix} P_{\mathbb{C}}(x) & \cdot \\ \cdot & \cdot \end{bmatrix}$$

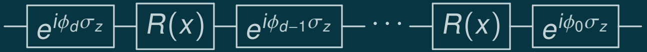
Indirect implementation



$$= \begin{bmatrix} P_{\mathbb{C}}(x) & \cdot & & \\ \cdot & \cdot & & \\ & & P^*_{\mathbb{C}}(x) & \cdot \\ & & \cdot & \cdot \end{bmatrix}$$

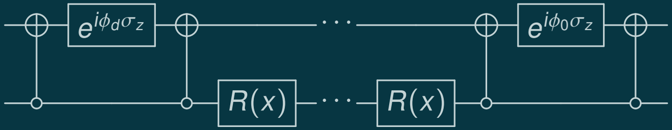
Implementing the real part of a polynomial map

Direct implementation



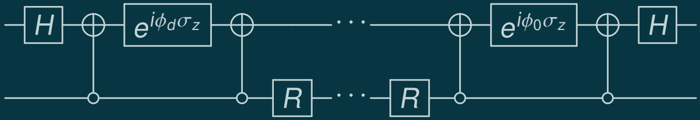
$$= \begin{bmatrix} P_{\mathbb{C}}(x) & \cdot \\ \cdot & \cdot \end{bmatrix}$$

Indirect implementation



$$= \begin{bmatrix} P_{\mathbb{C}}(x) & \cdot & & \\ \cdot & \cdot & & \\ & & P_{\mathbb{C}}^{*}(x) & \cdot \\ & & \cdot & \cdot \end{bmatrix}$$

Real implementation



$$= \begin{bmatrix} \Re[P_{\mathbb{C}}] & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

Generalisation to higher dimensions

1×1 case

$$\text{Input:} \begin{bmatrix} x & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \text{Modulation:} \begin{bmatrix} e^{i\phi} & \\ & e^{-i\phi} \end{bmatrix} \quad \text{Output:} \begin{bmatrix} P(x) & \cdot \\ \cdot & \cdot \end{bmatrix}$$

Generalisation to higher dimensions

1×1 case

$$\text{Input:}\begin{bmatrix} x & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \text{Modulation:}\begin{bmatrix} e^{i\phi} & \\ & e^{-i\phi} \end{bmatrix} \quad \text{Output:}\begin{bmatrix} P(x) & \cdot \\ \cdot & \cdot \end{bmatrix}$$

2×2 case (higher-dimensional case is similar)

Input unitary	Modulation	Output circuit
$\begin{bmatrix} x & \cdot & & \\ \cdot & \cdot & & \\ & & y & \cdot \\ & & \cdot & \cdot \end{bmatrix}$	$\begin{bmatrix} e^{i\phi} & & & \\ & e^{-i\phi} & & \\ & & e^{i\phi} & \\ & & & e^{-i\phi} \end{bmatrix}$	$\begin{bmatrix} P(x) & \cdot & & \\ \cdot & \cdot & & \\ & & P(y) & \cdot \\ & & \cdot & \cdot \end{bmatrix}$

Generalisation to higher dimensions

1×1 case

$$\text{Input:}\begin{bmatrix} x & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \text{Modulation:}\begin{bmatrix} e^{i\phi} & \\ & e^{-i\phi} \end{bmatrix} \quad \text{Output:}\begin{bmatrix} P(x) & \cdot \\ \cdot & \cdot \end{bmatrix}$$

2×2 case (higher-dimensional case is similar)

Input unitary	Modulation	Output circuit
$\begin{bmatrix} x & \cdot & & \\ \cdot & \cdot & & \\ & & y & \cdot \\ & & \cdot & \cdot \end{bmatrix}$	$\begin{bmatrix} e^{i\phi} & & & \\ & e^{-i\phi} & & \\ & & e^{i\phi} & \\ & & & e^{-i\phi} \end{bmatrix}$	$\begin{bmatrix} P(x) & \cdot & & \\ \cdot & \cdot & & \\ & & P(y) & \cdot \\ & & \cdot & \cdot \end{bmatrix}$
$\begin{bmatrix} x & & \cdot & \\ & y & & \cdot \\ \cdot & & \cdot & \\ & \cdot & & \cdot \end{bmatrix}$	$\begin{bmatrix} e^{i\phi} & & & \\ & e^{i\phi} & & \\ & & e^{-i\phi} & \\ & & & e^{-i\phi} \end{bmatrix}$	$\begin{bmatrix} P(x) & & \cdot & \\ & P(y) & & \cdot \\ \cdot & & \cdot & \\ & \cdot & & \cdot \end{bmatrix}$

Generalisation to higher dimensions

1×1 case

$$\text{Input:} \begin{bmatrix} x & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \text{Modulation:} \begin{bmatrix} e^{i\phi} & \\ & e^{-i\phi} \end{bmatrix} \quad \text{Output:} \begin{bmatrix} P(x) & \cdot \\ \cdot & \cdot \end{bmatrix}$$

2×2 case (higher-dimensional case is similar)

Input unitary	Modulation	Output circuit
$\begin{bmatrix} x & \cdot & & \\ \cdot & \cdot & & \\ & & y & \cdot \\ & & \cdot & \cdot \end{bmatrix}$	$\begin{bmatrix} e^{i\phi} & & & \\ & e^{-i\phi} & & \\ & & e^{i\phi} & \\ & & & e^{-i\phi} \end{bmatrix}$	$\begin{bmatrix} P(x) & \cdot & & \\ \cdot & \cdot & & \\ & & P(y) & \cdot \\ & & \cdot & \cdot \end{bmatrix}$
$\begin{bmatrix} x & & \cdot & \\ & y & & \cdot \\ \cdot & & \cdot & \\ & \cdot & & \cdot \end{bmatrix}$	$\begin{bmatrix} e^{i\phi} & & & \\ & e^{i\phi} & & \\ & & e^{-i\phi} & \\ & & & e^{-i\phi} \end{bmatrix}$	$\begin{bmatrix} P(x) & & \cdot & \\ & P(y) & & \cdot \\ \cdot & & \cdot & \\ & \cdot & & \cdot \end{bmatrix}$
$\begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix}$	$\begin{bmatrix} e^{i\phi} I & \\ & e^{-i\phi} I \end{bmatrix}$	$\begin{bmatrix} P(A) & \cdot \\ \cdot & \cdot \end{bmatrix}$

Direct implementation of HHL / the pseudoinverse

Singular value decomposition and pseudoinverse

Suppose $A = W\Sigma V^\dagger$ is a singular value decomposition.
Then the pseudoinverse of A is $A^+ = V\Sigma^+ W^\dagger$,

Direct implementation of HHL / the pseudoinverse

Singular value decomposition and pseudoinverse

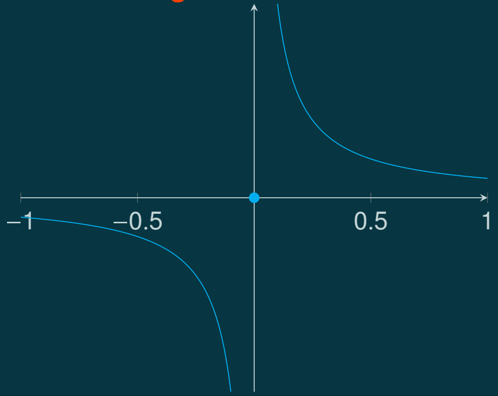
Suppose $A = W\Sigma V^\dagger$ is a singular value decomposition.
Then the pseudoinverse of A is $A^+ = V\Sigma^+ W^\dagger$, (note $A^\dagger = V\Sigma W^\dagger$)
where Σ^+ contains the inverses of the non-zero elements of Σ.

Direct implementation of HHL / the pseudoinverse

Singular value decomposition and pseudoinverse

Suppose $A = W\Sigma V^\dagger$ is a singular value decomposition.
Then the pseudoinverse of A is $A^+ = V\Sigma^+ W^\dagger$, (note $A^\dagger = V\Sigma W^\dagger$)
where Σ^+ contains the inverses of the non-zero elements of Σ.

Implementing the pseudoinverse using QSVT

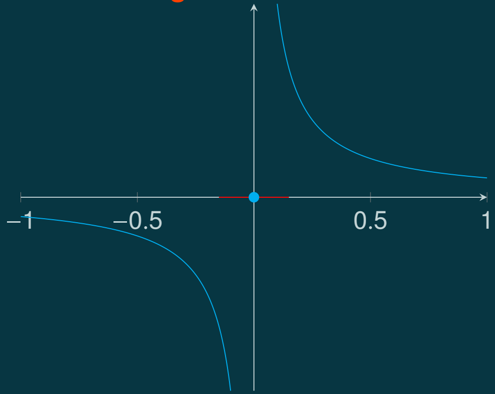


Direct implementation of HHL / the pseudoinverse

Singular value decomposition and pseudoinverse

Suppose $A = W\Sigma V^\dagger$ is a singular value decomposition.
Then the pseudoinverse of A is $A^+ = V\Sigma^+ W^\dagger$, (note $A^\dagger = V\Sigma W^\dagger$)
where Σ^+ contains the inverses of the non-zero elements of Σ.

Implementing the pseudoinverse using QSVT

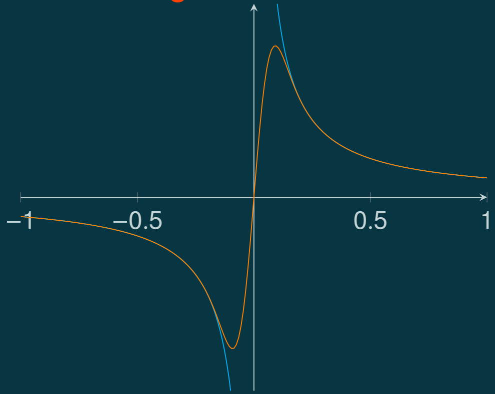


Direct implementation of HHL / the pseudoinverse

Singular value decomposition and pseudoinverse

Suppose $A = W\Sigma V^\dagger$ is a singular value decomposition.
Then the pseudoinverse of A is $A^+ = V\Sigma^+ W^\dagger$, (note $A^\dagger = V\Sigma W^\dagger$)
where Σ^+ contains the inverses of the non-zero elements of Σ.

Implementing the pseudoinverse using QSVT



Direct implementation of HHL / the pseudoinverse

Singular value decomposition and pseudoinverse

Suppose $A = W\Sigma V^\dagger$ is a singular value decomposition.
Then the pseudoinverse of A is $A^+ = V\Sigma^+ W^\dagger$, (note $A^\dagger = V\Sigma W^\dagger$)
where Σ^+ contains the inverses of the non-zero elements of Σ.

Implementing the pseudoinverse using QSVT

Suppose that U is an a-qubit block-encoding of A, and $\left\|A^+\right\| \leq \kappa$.

Direct implementation of HHL / the pseudoinverse

Singular value decomposition and pseudoinverse

Suppose $A = W\Sigma V^\dagger$ is a singular value decomposition.
Then the pseudoinverse of A is $A^+ = V\Sigma^+ W^\dagger$, (note $A^\dagger = V\Sigma W^\dagger$)
where Σ^+ contains the inverses of the non-zero elements of Σ.

Implementing the pseudoinverse using QSVT

Suppose that U is an a-qubit block-encoding of A, and $\left\|A^+\right\| \leq \kappa$.
By QSVT we can implement an ε-approximate block-encoding of

$$\frac{1}{2\kappa}A^+,$$

using $\mathcal{O}\left(\kappa \log\left(\frac{1}{\varepsilon}\right)\right)$ queries to U.

Direct implementation of HHL / the pseudoinverse

Singular value decomposition and pseudoinverse

Suppose $A = W\Sigma V^\dagger$ is a singular value decomposition.
Then the pseudoinverse of A is $A^+ = V\Sigma^+ W^\dagger$, (note $A^\dagger = V\Sigma W^\dagger$)
where Σ^+ contains the inverses of the non-zero elements of Σ.

Implementing the pseudoinverse using QSVT

Suppose that U is an a-qubit block-encoding of A, and $\left\|A^+\right\| \leq \kappa$.
By QSVT we can implement an ε-approximate block-encoding of

$$\frac{1}{2\kappa}A^+,$$

using $\mathcal{O}\left(\kappa \log\left(\frac{1}{\varepsilon}\right)\right)$ queries to U. For the corresponding approximating polynomial, see, e.g., the work of Childs, Kothari and Somma (2015).

Singular vector transformation and projection

New result: Singular vector transformation

Given a unitary U, and projectors $\widetilde{\Pi}, \Pi$, such that

$$A = \widetilde{\Pi} U \Pi = \sum_{i=1}^{k} \varsigma_i |\phi_i\rangle\langle\psi_i|$$

is a singular value decomposition.

Singular vector transformation and projection

New result: Singular vector transformation

Given a unitary U, and projectors $\widetilde{\Pi}, \Pi$, such that

$$A = \widetilde{\Pi} U \Pi = \sum_{i=1}^{k} \varsigma_i |\phi_i\rangle\langle\psi_i|$$

is a singular value decomposition. Transform one copy of a quantum state

$$|\psi\rangle = \sum_{i=i}^{k} \alpha_i |\psi_i\rangle \quad \text{to} \quad |\phi\rangle = \sum_{i=i}^{k} \alpha_i |\phi_i\rangle.$$

Singular vector transformation and projection

New result: Singular vector transformation

Given a unitary U, and projectors $\widetilde{\Pi}, \Pi$, such that

$$A = \widetilde{\Pi} U \Pi = \sum_{i=1}^{k} \varsigma_i |\phi_i\rangle\langle\psi_i|$$

is a singular value decomposition. Transform one copy of a quantum state

$$|\psi\rangle = \sum_{i=i}^{k} \alpha_i |\psi_i\rangle \quad \text{to} \quad |\phi\rangle = \sum_{i=i}^{k} \alpha_i |\phi_i\rangle.$$

If $\varsigma_i \geq \delta$ for all $0 \neq \alpha_i$, we can ε-apx. using QSVT with compl. $\mathcal{O}\left(\frac{1}{\delta}\log\left(\frac{1}{\varepsilon}\right)\right)$.

Singular vector transformation and projection

New result: Singular vector transformation

Given a unitary U, and projectors $\widetilde{\Pi}, \Pi$, such that

$$A = \widetilde{\Pi} U \Pi = \sum_{i=1}^{k} \varsigma_i |\phi_i\rangle\langle\psi_i|$$

is a singular value decomposition. Transform one copy of a quantum state

$$|\psi\rangle = \sum_{i=i}^{k} \alpha_i |\psi_i\rangle \quad \text{to} \quad |\phi\rangle = \sum_{i=i}^{k} \alpha_i |\phi_i\rangle.$$

If $\varsigma_i \geq \delta$ for all $0 \neq \alpha_i$, we can ε-apx. using QSVT with compl. $\mathcal{O}\left(\frac{1}{\delta}\log\left(\frac{1}{\varepsilon}\right)\right)$.

Fixed-point and oblivious amplitude ampl. [Yoder et al., Berry et al.]

Amplitude amplification problem: Given U such that

$$U|\psi_{\text{in}}\rangle = \sqrt{p}|0\rangle\left|\psi_{\text{good}}\right\rangle + \sqrt{1-p}|1\rangle|\psi_{\text{bad}}\rangle, \quad \text{prepare } \left|\psi_{\text{good}}\right\rangle.$$

Singular vector transformation and projection

New result: Singular vector transformation

Given a unitary U, and projectors $\widetilde{\Pi}, \Pi$, such that

$$A = \widetilde{\Pi} U \Pi = \sum_{i=1}^{k} \varsigma_i |\phi_i\rangle\langle\psi_i|$$

is a singular value decomposition. Transform one copy of a quantum state

$$|\psi\rangle = \sum_{i=i}^{k} \alpha_i |\psi_i\rangle \quad \text{to} \quad |\phi\rangle = \sum_{i=i}^{k} \alpha_i |\phi_i\rangle.$$

If $\varsigma_i \geq \delta$ for all $0 \neq \alpha_i$, we can ε-apx. using QSVT with compl. $\mathcal{O}\left(\frac{1}{\delta}\log\left(\frac{1}{\varepsilon}\right)\right)$.

Fixed-point and oblivious amplitude ampl. [Yoder et al., Berry et al.]

Amplitude amplification problem: Given U such that

$$U|\psi_{\text{in}}\rangle = \sqrt{p}|0\rangle\left|\psi_{\text{good}}\right\rangle + \sqrt{1-p}|1\rangle|\psi_{\text{bad}}\rangle, \quad \text{prepare } \left|\psi_{\text{good}}\right\rangle.$$

Note that $(|0\rangle\langle 0| \otimes I)U(|\psi_{\text{in}}\rangle\langle\psi_{\text{in}}|) = \sqrt{p}|0,\psi_{\text{good}}\rangle\langle\psi_{\text{in}}|$; we can apply QSVT.

Optimal block-Hamiltonian simulation

Suppose that H is given as an a-qubit block-encoding, i.e., $U = \begin{bmatrix} H & \cdot \\ \cdot & \cdot \end{bmatrix}$.

Optimal block-Hamiltonian simulation

Suppose that H is given as an a-qubit block-encoding, i.e., $U = \begin{bmatrix} H & . \\ . & . \end{bmatrix}$.

Complexity of block-Hamiltonians simulation [Low & Chuang (2016)]

Given $t, \varepsilon > 0$, implement a unitary U', which is ε close to e^{itH}. Can be achieved with query complexity

$$\mathcal{O}\left(t + \log(1/\varepsilon)\right).$$

Gate complexity is $\mathcal{O}\left(a\right)$ times the above.

Optimal block-Hamiltonian simulation

Suppose that H is given as an a-qubit block-encoding, i.e., $U = \begin{bmatrix} H & \cdot \\ \cdot & \cdot \end{bmatrix}$.

Complexity of block-Hamiltonians simulation [Low & Chuang (2016)]

Given $t, \varepsilon > 0$, implement a unitary U', which is ε close to e^{itH}. Can be achieved with query complexity

$$\mathcal{O}\left(t + \log(1/\varepsilon)\right).$$

Gate complexity is $\mathcal{O}(a)$ times the above.

Proof sketch

Approximate to ε-precision $\sin(tx)$ and $\cos(tx)$ with polynomials of degree as above. Then use QSVT and combine even/odd parts.

Optimal complexity

$$\Theta\left(t + \frac{\log(1/\varepsilon)}{\log(e + \log(1/\varepsilon)/t)}\right)$$

Optimal block-Hamiltonian simulation

Suppose that H is given as an a-qubit block-encoding, i.e., $U = \begin{bmatrix} H & . \\ . & . \end{bmatrix}$.

Complexity of block-Hamiltonians simulation [Low & Chuang (2016)]

Given $t, \varepsilon > 0$, implement a unitary U', which is ε close to e^{itH}. Can be achieved with query complexity

$$\mathcal{O}\left(t + \log(1/\varepsilon)\right).$$

Gate complexity is $\mathcal{O}\left(a\right)$ times the above.

Proof sketch

Approximate to ε-precision $\sin(tx)$ and $\cos(tx)$ with polynomials of degree as above. Then use QSVT and combine even/odd parts.

Optimal complexity

$$\Theta\left(t + \frac{\log(1/\varepsilon)}{\log(e + \log(1/\varepsilon)/t)}\right)$$ cf. density matrix exp. $\Theta(t^2/\varepsilon)$ Lloyd et al., Kimmel et al.]

Distributional property testing [Bravy et al. ’09] [G. and Li ’19]

Suppose we can implement ”quantum sampling“: $U_p : |0\rangle \mapsto \sum_i \sqrt{p_i}|\phi_i\rangle|i\rangle$

Distributional property testing [Bravy et al. ’09] [G. and Li ’19]

Suppose we can implement ”quantum sampling“: $U_p : |0\rangle \mapsto \sum_i \sqrt{p_i}|\phi_i\rangle|i\rangle$

How can we efficiently compute, e.g., the entropy of (p_i)?

Apply a block-encoding of a map $|i\rangle \mapsto \sqrt{\log(p_i)}|i\rangle$ to the state, then estimate the amplitude.

Distributional property testing [Bravy et al. '09] [G. and Li '19]

Suppose we can implement "quantum sampling": $U_p : |0\rangle \mapsto \sum_i \sqrt{p_i}|\phi_i\rangle|i\rangle$

How can we efficiently compute, e.g., the entropy of (p_i)?

Apply a block-encoding of a map $|i\rangle \mapsto \sqrt{\log(p_i)}|i\rangle$ to the state, then estimate the amplitude.
We need a block-encoding of $\sum_i \sqrt{p_i}|i\rangle\langle i|$, and then transform singular values?

Distributional property testing [Bravy et al. '09] [G. and Li '19]

Suppose we can implement "quantum sampling": $U_p : |0\rangle \mapsto \sum_i \sqrt{p_i}|\phi_i\rangle|i\rangle$

How can we efficiently compute, e.g., the entropy of (p_i)?

Apply a block-encoding of a map $|i\rangle \mapsto \sqrt{\log(p_i)}|i\rangle$ to the state, then estimate the amplitude.
We need a block-encoding of $\sum_i \sqrt{p_i}|i\rangle\langle i|$, and then transform singular values?
Observation: a block encoding of $\sum_i \sqrt{p_i}|\tilde{\phi}_i\rangle\langle i|$ suffices.

Distributional property testing [Bravy et al. ’09] [G. and Li ’19]

Suppose we can implement ”quantum sampling“: $U_p : |0\rangle \mapsto \sum_i \sqrt{p_i}|\phi_i\rangle|i\rangle$

How can we efficiently compute, e.g., the entropy of (p_i)?

Apply a block-encoding of a map $|i\rangle \mapsto \sqrt{\log(p_i)}|i\rangle$ to the state, then estimate the amplitude.
We need a block-encoding of $\sum_i \sqrt{p_i}|i\rangle\langle i|$, and then transform singular values?
Observation: a block encoding of $\sum_i \sqrt{p_i}|\tilde{\phi}_i\rangle\langle i|$ suffices.

The same technique works for density operators!

Distributional property testing [Bravy et al. ’09] [G. and Li ’19]

Suppose we can implement ”quantum sampling“: $U_p : |0\rangle \mapsto \sum_i \sqrt{p_i}|\phi_i\rangle|i\rangle$

How can we efficiently compute, e.g., the entropy of (p_i)?

Apply a block-encoding of a map $|i\rangle \mapsto \sqrt{\log(p_i)}|i\rangle$ to the state, then estimate the amplitude.
We need a block-encoding of $\sum_i \sqrt{p_i}|i\rangle\langle i|$, and then transform singular values?
Observation: a block encoding of $\sum_i \sqrt{p_i}|\tilde{\phi}_i\rangle\langle i|$ suffices.

The same technique works for density operators!
Purified access $U_\rho : |0\rangle \mapsto \sum_i \sqrt{p_i}|\phi_i\rangle|\psi_i\rangle$, where $\rho = \sum_i p_i|\psi_i\rangle\langle\psi_i|$

An intuitive lower bound

Lower bound on eigenvalue transformation

Suppose that U is a block-encoding of a Hermitian matrix H from a family of operators. Let $f\colon [-1,1] \to \mathbb{C}$, then implementing a block-encoding of $f(H)$ requires at least $\left\| \frac{df}{dx} \right\|_I$ uses of U, if $I \subseteq [-\frac{1}{2}, \frac{1}{2}]$ is an interval of potential eigenvalues of H.

An intuitive lower bound

Lower bound on eigenvalue transformation

Suppose that U is a block-encoding of a Hermitian matrix H from a family of operators. Let $f\colon [-1,1] \to \mathbb{C}$, then implementing a block-encoding of $f(H)$ requires at least $\left\| \frac{df}{dx} \right\|_I$ uses of U, if $I \subseteq [-\frac{1}{2}, \frac{1}{2}]$ is an interval of potential eigenvalues of H.

Proof sketch

The proof is based on an elementary argument about distinguishability of unitary operators.

An intuitive lower bound

Lower bound on eigenvalue transformation

Suppose that U is a block-encoding of a Hermitian matrix H from a family of operators. Let $f\colon [-1,1] \to \mathbb{C}$, then implementing a block-encoding of $f(H)$ requires at least $\left\|\frac{df}{dx}\right\|_I$ uses of U, if $I \subseteq [-\frac{1}{2}, \frac{1}{2}]$ is an interval of potential eigenvalues of H.

Proof sketch

The proof is based on an elementary argument about distinguishability of unitary operators.

Optimality of pseudoinverse implementation

$$\text{Let } I := \left[\frac{1}{\kappa}, \frac{1}{2}\right] \text{ and let } f(x) := \frac{1}{\kappa x}, \text{ then } \left.\frac{df}{dx}\right|_{\frac{1}{\kappa}} = -\kappa.$$

Thus our implementation is optimal up to the $\log(1/\varepsilon)$ factor.

Summarizing the various speed-ups

Speed-up	Source of speed-up	Examples of algorithms
Exponential	Dimensionality of the Hilbert space	Hamiltonian simulation
	Precise polynomial approximations	Improved HHL algorithm
Quadratic	Singular value = square root of probability	Grover search
	Singular values are easier to distinguish	Amplitude estimation
	Close-to-1 singular values are more flexible	Quantum walks

Summarizing the various speed-ups

Speed-up	Source of speed-up	Examples of algorithms
Exponential	Dimensionality of the Hilbert space	Hamiltonian simulation
	Precise polynomial approximations	Improved HHL algorithm
Quadratic	Singular value = square root of probability	Grover search
	Singular values are easier to distinguish	Amplitude estimation
	Close-to-1 singular values are more flexible	Quantum walks

Some more applications

- Quantum walks, fast QMA amplification, fast quantum OR lemma
- Quantum Machine learning: PCA, principal component regression
- “Non-commutative measurements” (for ground state preparation)
- Fractional queries
- $\vdots$