Grand Unification of Quantum Algorithms

András Gilyén
Alfréd Rényi Institute of Mathematics
Budapest, Hungary

Quantum Computing Summer School, Physikzentrum Bad Honnef, Germany
2022 August 14-19
Quantum algorithm design
Quantum algorithm design

Many quantum algorithms have a common structure!
A bird’s eye view on quantum linear algebra

Motivating example - the quantum matrix inversion (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


In HHL $f(x) = 1/x$. Use Singular Value Transformation to approximate it!

A bird’s eye view on quantum linear algebra

Motivating example - the quantum matrix inversion (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 \(\propto A^{-1}|b\rangle\).
A bird’s eye view on quantum linear algebra

Motivating example - the quantum matrix inversion (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

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

In HHL \( f(x) = \frac{1}{x} \). Use Singular Value Transformation to approximate it!
A bird’s eye view on quantum linear algebra

Motivating example - the quantum matrix inversion (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

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

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

More examples

- Optimal Hamiltonian simulation [Low et al.], quantum walks [Szegedy]
- Fixed point [Yoder et al.] and oblivious amplitude amplification [Berry et al.]
- HHL, regression [Chakraborty et al.], SDPs & LPs [Brandão et al.], ML [Kerendis et al.]
Block-encoding

A way to represent large matrices on a quantum computer efficiently

\[ U = \begin{bmatrix} A & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \iff A = (\langle 0| \otimes I) U (|0\rangle \otimes I). \]
Block-encoding

A way to represent large matrices on a quantum computer efficiently

\[ U = \begin{bmatrix} A \ \cdot \ \cdot \end{bmatrix} \iff A = (\langle 0^a \otimes I \rangle \ U \ (|0^a \rangle \otimes I)) . \]

Any complex matrix \( A \) with operator norm \( \|A\| \leq 1 \) can be block-encoded.
Block-encoding

A way to represent large matrices on a quantum computer efficiently

\[ U = \begin{bmatrix} A & \cdots \\ \cdots & \ddots \end{bmatrix} \iff A = (\langle 0 |^a \otimes I) \ U \ (|0 \rangle^a \otimes I) \ . \]

Any complex matrix \( A \) with operator norm \( \| A \| \leq 1 \) can be block-encoded.

One can efficiently construct block-encodings of
Block-encoding

A way to represent large matrices on a quantum computer efficiently

\[ U = \begin{bmatrix} A & \ddots \\ \vdots & \ddots \\ \end{bmatrix} \iff A = (\langle 0 |^a \otimes I) \ U \ (|0 \rangle^a \otimes I). \]

Any complex matrix \( A \) with operator norm \( \|A\| \leq 1 \) can be block-encoded.

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

\[ U = \begin{bmatrix} A & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \iff A = (\langle 0 \rangle^a \otimes I) U (\lvert 0 \rangle^a \otimes I). \]

Any complex matrix \( A \) with operator norm \( \| A \| \leq 1 \) can be block-encoded.

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

\[ U = \begin{bmatrix} A & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \iff A = (\langle 0|^{a} \otimes I) U (|0\rangle^{a} \otimes I) . \]

Any complex matrix \( A \) with operator norm \( \|A\| \leq 1 \) can be block-encoded.

**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

\[ U = \begin{bmatrix} A \end{bmatrix} \iff A = (\langle 0^a \otimes 1 \rangle) U (|0^a \rangle \otimes 1). \]

Any complex matrix \( A \) with operator norm \( \|A\| \leq 1 \) can be block-encoded.

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 \( \rho \) given a unitary preparing its purification.
Block-encoding

A way to represent large matrices on a quantum computer efficiently

\[
U = \begin{bmatrix} A & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\end{bmatrix} \iff A = (\langle 0|^a \otimes I) U (|0\rangle^a \otimes I).
\]

Any complex matrix \( A \) with operator norm \( \|A\| \leq 1 \) can be block-encoded.

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 \( \rho \) given a unitary preparing its purification.
- a POVM operator \( M \) given we can sample from the rand.var.: \( \text{Tr}(\rho M) \),
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" $R$: $|0\rangle|0\rangle|i\rangle \rightarrow |0\rangle \sum_k (\sqrt{A_{ik}})^{\ast} \sqrt{s} |i\rangle |k\rangle + |1\rangle |i\rangle |\text{garbage}\rangle$,

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

They form a block-encoding of $A/s$: $\langle 0|\langle 0|\langle i| R^\dagger C |0\rangle|0\rangle|j\rangle = (R|0\rangle|0\rangle|i\rangle) \dagger \cdot (C|0\rangle|0\rangle|j\rangle) = \begin{pmatrix} \sum_k (\sqrt{A_{ik}})^{\ast} \sqrt{s} \end{pmatrix} \dagger \begin{pmatrix} \sum_\ell \sqrt{A_{\ell j}} \sqrt{s} \end{pmatrix}$.
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"

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

and "columns":

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

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

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"

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

and "columns"

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

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“

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

and ”columns“

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

They form a block-encoding of $A/s$:

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

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

They form a block-encoding of $A/s$:

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

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

and "columns"

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

They form a block-encoding of $A/s$:

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

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 $U = \sum |i\rangle\langle i| \otimes U_i$, 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 $\sum_p p_i U_i$.

In particular if $(\langle 0 | \otimes I) U_i (|0\rangle \otimes I) = A_i$, then it is a block-encoding of $\sum p_i A_i$. 

Efficient matrix arithmetics

Implementing arithmetic operations on block-encoded matrices

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

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 $\sum_i p_i A_i$. 

5 / 20
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

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

$$\left( P^\dagger \otimes I \right) U \left( P \otimes I \right)$$

is a block-encoding of $\sum_i p_i U_i$.

In particular if

$$\left( \langle 0 | \otimes I \right) U_i \left( |0\rangle \otimes I \right) = A_i,$$

then it is a block-encoding of $\sum_i p_i A_i$. 

5 / 20
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 \( P : [-1, 1] \rightarrow [-1, 1] \) be a degree-\( d \) odd polynomial map.
Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

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

$$U = \begin{bmatrix} A & \cdots \\ \vdots & \ddots \end{bmatrix} = \begin{bmatrix} \sum_i s_i |w_i \rangle \langle v_i| & \cdots \end{bmatrix}$$
Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

Let \( P : [-1, 1] \rightarrow [-1, 1] \) be a degree-\( d \) odd polynomial map. Suppose that

\[
U = \begin{bmatrix} A \\ \\ \end{bmatrix} = \begin{bmatrix} \sum_i s_i |w_i \rangle \langle v_i| \\ \end{bmatrix} \implies U_\Phi = \begin{bmatrix} \sum_i P(s_i) |w_i \rangle \langle v_i| \\ \end{bmatrix},
\]

where \( U_\Phi \) is efficiently computable and \( U_\Phi \) is the following circuit:

Alternating phase modulation sequence

\[
U_\Phi := H e^{-i \phi_1} \sigma_z e^{-i \phi_2} \sigma_z \cdots e^{-i \phi_d} \sigma_z H U U^\dagger \cdots \cdots
\]

Similar result holds for even polynomials.
Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

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

$$U = \sum_i \varphi_i |w_i v_i| \rightarrow U_{\Phi} = \sum_i P(\varphi_i) |w_i v_i|,$$

where $\Phi(P) \in \mathbb{R}^d$ is efficiently computable and $U_{\Phi}$ is the following circuit:
Quantum Singular Value Transformation (QSVT)

Our main theorem about QSVT

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

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

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

Alternating phase modulation sequence $U_\Phi :=$

\[
\begin{array}{ccccccccc}
\text{H} & e^{-i\phi_1 \sigma_z} & \text{H} & e^{-i\phi_2 \sigma_z} & \cdots & e^{-i\phi_d \sigma_z} & \text{H} \\
\mid 0 \rangle^\otimes a & \Downarrow U & \Downarrow U^\dagger & \Downarrow \cdots & \Downarrow \cdots & \Downarrow \cdots & \Downarrow \cdots \\
\end{array}
\]
Quantum Singular Value Transformation (QSVT)

**Our main theorem about QSVT**

Let \( P: [-1, 1] \rightarrow [-1, 1] \) be a degree-\( d \) odd polynomial map. Suppose that

\[
U = \begin{bmatrix} A & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} = \left[ \sum_i \varsigma_i |w_i \rangle \langle v_i| \right] \quad \Rightarrow \quad U_\Phi = \left[ \sum_i P(\varsigma_i) |w_i \rangle \langle v_i| \right],
\]

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

**Alternating phase modulation sequence** \( U_\Phi := \)

\[\begin{array}{cccccccc}
H & e^{-i \phi_1 \sigma_z} & e^{-i \phi_2 \sigma_z} & \cdots & e^{-i \phi_d \sigma_z} & H \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
|0\rangle^\otimes a & U & U^\dagger & \cdots & \cdots & \cdots
\end{array}\]

Similar result holds for even polynomials.
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 $A^+ = V \Sigma^+ W^\dagger$, where $\Sigma^+$ contains the inverses of the non-zero elements of $\Sigma$. 
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 \( \Sigma^+ \) contains the inverses of the non-zero elements of \( \Sigma \).
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 $\Sigma^+$ contains the inverses of the non-zero elements of $\Sigma$.

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 $\Sigma^+$ contains the inverses of the non-zero elements of $\Sigma$.

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 $\Sigma^+$ contains the inverses of the non-zero elements of $\Sigma$.

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 \( \Sigma^+ \) contains the inverses of the non-zero elements of \( \Sigma \).

Implementing the pseudoinverse using QSVT

Degree / complexity: \( O(\kappa \log \left( \frac{1}{\varepsilon} \right)) \)
Quantum walks and Hermitian matrices

Connection to Szegedy quantum walks

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

Markov chain: \( M \); Updates: \( U = \begin{bmatrix} M & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \); Walk: \( W^n = \begin{bmatrix} T_{2n}(M) & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \).

(\( T_d \) 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

- **Markov chain**: $M$
- **Updates**: $U = \begin{bmatrix} M & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$
- **Walk**: $W^n = \begin{bmatrix} T_{2n}(M) & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$

($T_d$ 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

Markov chain: $M$; Updates: $U = \begin{bmatrix} M & \cdots \\ \vdots & \ddots \end{bmatrix}$; Walk: $W^n = \begin{bmatrix} T_{2n}(M) & \cdots \\ \vdots & \ddots \end{bmatrix}$.

($T_d$ 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

Markov chain: $M$; Updates: $U = \begin{bmatrix} M & \cdots \\ \vdots & \ddots \end{bmatrix}$; Walk: $W^n = \begin{bmatrix} T_{2n}(M) & \cdots \\ \vdots & \ddots \end{bmatrix}$.

($T_d$ 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

Markov chain: $M$; Updates: $U = \begin{bmatrix} M & \cdot & \cdot \end{bmatrix}$; Walk: $W^n = \begin{bmatrix} T_{2n}(M) & \cdot & \cdot \end{bmatrix}$.

($T_d$ 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

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

Markov chain: $M$; Updates: $U = \begin{bmatrix} M & \cdot & \cdot \\ & \ddots & \cdot \\ & & \ddots & \cdot \\ \end{bmatrix}$; Walk: $W^n = \begin{bmatrix} T_{2n}(M) & \cdot & \cdot \\ & \ddots & \cdot \\ & & \ddots & \cdot \\ \end{bmatrix}$.

($T_d$ 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

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

Proof: $x^t$ can be $\varepsilon$-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 $P : [-1, 1] \to [-1, 1]$ be a degree-$d$ even/odd polynomial map.

The special case of Hermitian matrices

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

Let $P : [-1, 1] \rightarrow [-1, 1]$ 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 $P : [-1, 1] \rightarrow [-1, 1]$ 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 $P : [-1, 1] \rightarrow [-\frac{1}{2}, \frac{1}{2}]$ be a degree-$d$ polynomial map. Suppose that $U$ is an $a$-qubit block-encoding of a Hermitian matrix $H$. We can implement $U' = P(H)$ using $d$ times $U$ and $U^\dagger$, 1 controlled $U$, and $O(\text{ad})$ extra two-qubit gates.

Proof: let $P_{\text{even}}(x) := P(x) + P(-x)$ and $P_{\text{odd}}(x) := P(x) - P(-x)$ then $P(H) = \frac{1}{2}(P_{\text{even}}(H) + P_{\text{odd}}(H))$ implement using QSVT + LCU.
The special case of Hermitian matrices

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

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

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

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

Proof: let $P_{\text{even}}(x) := P(x) + P(-x)$ and $P_{\text{odd}}(x) := P(x) - P(-x)$ then

$$P(H) = \frac{1}{2}(P_{\text{even}}(H) + P_{\text{odd}}(H)) \quad \text{implement using QSVT + LCU}$$
Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using $\sigma_z$ phases?

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

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

\[
(\ast) = i^d \begin{pmatrix}
P_C(x)Q_C(x) \\
Q^*_C(x)P^*_C(x)
\end{pmatrix},
\]

where $P_C$, $Q_C \in \mathbb{C}[x]$ are such that

(i) $\deg(P_C) \leq d$ and $\deg(Q_C) \leq d - 1$,

(ii) $P_C$ has parity-$\left(\frac{d}{2}\right)$ and $Q_C$ has parity-$\left(\frac{d - 1}{2}\right)$,

(iii) $\forall x \in [-1, 1]: |P_C(x)|^2 + (1 - x^2)|Q_C(x)|^2 = 1$. 


Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using $\sigma_z$ phases?

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

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

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

$$(\ast) = i^d \begin{bmatrix} P_C(x) & Q_C(x) \cdot i \sqrt{1-x^2} \\ Q_C^*(x) & P_C^*(x) \cdot i \sqrt{1-x^2} \end{bmatrix},$$

where $P_C, Q_C \in \mathbb{C}[x]$ are such that

(i) $\deg(P_C) \leq d$ and $\deg(Q_C) \leq d - 1$, and

(ii) $P_C$ has parity-$(d \mod 2)$ and $Q_C$ has parity-$(d - 1 \mod 2)$, and

(iii) $\forall x \in [-1, 1]: |P_C(x)|^2 + (1-x^2) |Q_C(x)|^2 = 1.$
Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using $\sigma_z$ 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

$$(*) = i^d \begin{bmatrix} P_C(x) & Q_C(x) i \sqrt{1-x^2} \\ Q_C^*(x) i \sqrt{1-x^2} & P_C^*(x) \end{bmatrix},$$

where $P_C, Q_C \in \mathbb{C}[x]$ are such that
Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using $\sigma_z$ phases?

$$R(x) := \begin{bmatrix} x & -\sqrt{1-x^2} \\ -\sqrt{1-x^2} & -x \end{bmatrix}$$

$e^{i\phi_0 \sigma_z} R(x) e^{i\phi_1 \sigma_z} \cdots R(x) e^{i\phi_d \sigma_z} = (*)$?

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

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

$$(*) = i^d \begin{bmatrix} P_C(x) & Q_C(x)i \sqrt{1-x^2} \\ Q_C^*(x)i \sqrt{1-x^2} & P_C^*(x) \end{bmatrix},$$

where $P_C, Q_C \in \mathbb{C}[x]$ are such that

(i) $\deg(P_C) \leq d$ and $\deg(Q_C) \leq d - 1$, and
Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using $\sigma_z$ 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

$$(*) = i^d \begin{bmatrix} P_C(x) & Q_C(x) i \sqrt{1-x^2} \\ Q_C^*(x) i \sqrt{1-x^2} & P_C^*(x) \end{bmatrix},$$

where $P_C, Q_C \in \mathbb{C}[x]$ are such that

(i) $\deg(P_C) \leq d$ and $\deg(Q_C) \leq d - 1$, and

(ii) $P_C$ has parity-$(d \mod 2)$ and $Q_C$ has parity-$(d - 1 \mod 2)$, and
Quantum signal processing & proof sketch of QSVT

Single qubit quantum control using $\sigma_z$ 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

$$(*) = i^d \begin{bmatrix} P_C(x) & Q_C(x) i \sqrt{1-x^2} \\ Q_C^*(x) \sqrt{1-x^2} & P_C^*(x) \end{bmatrix},$$

where $P_C, Q_C \in \mathbb{C}[x]$ are such that

(i) $\text{deg}(P_C) \leq d$ and $\text{deg}(Q_C) \leq d - 1$, and

(ii) $P_C$ has parity-$(d \mod 2)$ and $Q_C$ has parity-$(d - 1 \mod 2)$, and

(iii) $\forall x \in [-1, 1]: |P_C(x)|^2 + (1-x^2)|Q_C(x)|^2 = 1.$
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

\[
\prod_{j=1}^{d} \left( R(x) e^{i\phi_j \sigma_z} \right) = \begin{bmatrix} P_C(x) & \cdots \\ \cdots & \cdots \end{bmatrix},
\]

where \( \mathbb{R}[P_C] = P \) 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

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

where $\mathbb{R}[P_{\mathbb{C}}] = P$ 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

$$\prod_{j=1}^{d} (R(x)e^{i\phi_j \sigma_z}) = [P_{\mathbb{C}}(x) \ldots],$$

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

\[ e^{i\phi_d \sigma_z} R(x) e^{i\phi_{d-1} \sigma_z} \cdots R(x) e^{i\phi_0 \sigma_z} = \begin{bmatrix} P_C(x) \end{bmatrix} \]

Real implementation

\[ H e^{i\phi_d \sigma_z} R(x) e^{i\phi_{d-1} \sigma_z} \cdots R(x) e^{i\phi_0 \sigma_z} = \begin{bmatrix} \Re \left[ P_C(x) \right] \end{bmatrix} \]
Implementing the real part of a polynomial map

Direct implementation

\[
e^{i\phi_d \sigma_z} \quad R(x) \quad e^{i\phi_{d-1} \sigma_z} \quad \cdots \quad R(x) \quad e^{i\phi_0 \sigma_z} = \begin{bmatrix} P_C(x) \end{bmatrix}
\]

Indirect implementation

\[
e^{i\phi_d \sigma_z} \quad \cdots \quad e^{i\phi_0 \sigma_z} = \begin{bmatrix} P_C(x) \end{bmatrix} \quad \begin{bmatrix} P^*_C(x) \end{bmatrix}
\]
Implementing the real part of a polynomial map

Direct implementation

\[ e^{i\phi_d \sigma_z} R(x) e^{i\phi_{d-1} \sigma_z} \cdots R(x) e^{i\phi_0 \sigma_z} = \begin{bmatrix} P_C(x) \end{bmatrix} \]

Indirect implementation

\[ e^{i\phi_d \sigma_z} \cdots e^{i\phi_0 \sigma_z} R(x) \cdots R(x) = \begin{bmatrix} P_C(x) \end{bmatrix} \begin{bmatrix} \bar{P}_C(x) \end{bmatrix} \]

Real implementation

\[ H e^{i\phi_d \sigma_z} \cdots e^{i\phi_0 \sigma_z} H = \begin{bmatrix} \Re[P_C] \end{bmatrix} \begin{bmatrix} \Re[P_C] \end{bmatrix} \]

12 / 20
Generalisation to higher dimensions

$1 \times 1$ case

Input: $\begin{bmatrix} x \end{bmatrix}$

Modulation: $\begin{bmatrix} e^{i\phi} & e^{-i\phi} \end{bmatrix}$

Output: $\begin{bmatrix} P(x) \end{bmatrix}$
Generalisation to higher dimensions

1 × 1 case

Input: \[
\begin{bmatrix}
  x \\
  . \\
  . \\
\end{bmatrix}
\]
Modulation: \[
\begin{bmatrix}
  e^{i\phi} \\
  e^{-i\phi}
\end{bmatrix}
\]
Output: \[
\begin{bmatrix}
  P(x) \\
  . \\
  . \\
\end{bmatrix}
\]

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

<table>
<thead>
<tr>
<th>Input unitary</th>
<th>Modulation</th>
<th>Output circuit</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
  x \\
  . \\
  . \\
  y \\
  . \\
  . \\
\end{bmatrix}
\] | \[
\begin{bmatrix}
  e^{i\phi} \\
  e^{-i\phi} \\
  e^{i\phi} \\
  e^{-i\phi}
\end{bmatrix}
\] | \[
\begin{bmatrix}
  P(x) \\
  . \\
  . \\
  P(y) \\
  . \\
  . \\
\end{bmatrix}
\] |
Generalisation to higher dimensions

1 × 1 case

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

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

Input unitary
\[ \begin{bmatrix} x & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ y & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \]
Modulation
\[ \begin{bmatrix} e^{i\phi} & e^{-i\phi} \\ e^{i\phi} & e^{-i\phi} \\ e^{i\phi} & e^{-i\phi} \\ e^{i\phi} & e^{-i\phi} \end{bmatrix} \]
Output circuit
\[ \begin{bmatrix} P(x) & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ P(y) & \cdot & \cdot & \cdot \\ P(x) & P(y) & \cdot & \cdot \end{bmatrix} \]
Generalisation to higher dimensions

1 × 1 case

<table>
<thead>
<tr>
<th>Input:</th>
<th>Modulation:</th>
<th>Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>[x . .]</td>
<td>[e^{i\phi} \quad e^{-i\phi}]</td>
<td>[P(x) . .]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Input unitary</th>
<th>Modulation</th>
<th>Output circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[x . .]</td>
<td>[e^{i\phi} \quad e^{-i\phi}]</td>
<td>[P(x) . .]</td>
</tr>
<tr>
<td>[y . .]</td>
<td>[e^{i\phi} \quad e^{-i\phi}]</td>
<td>[P(y) . .]</td>
</tr>
<tr>
<td>[x . y . .]</td>
<td>[e^{i\phi} \quad e^{-i\phi}]</td>
<td>[P(x) \quad P(y) . .]</td>
</tr>
<tr>
<td>[A . .]</td>
<td>[e^{i\phi} I \quad e^{-i\phi} I]</td>
<td>[P(A) . .]</td>
</tr>
</tbody>
</table>
The language class QMA

The language $L$ belongs to the class QMA if for every input length $|x|$ there exists a quantum verifier $V_{|x|}$, and numbers $0 \leq b_{|x|} < a_{|x|} \leq 1$ satisfying $\frac{1}{a_{|x|}-b_{|x|}} = O\left(\text{poly}\left(|x|\right)\right)$, such that for all $x \in L$ there exists a witness $|\psi\rangle$ such that upon measuring the state $V_{|x|}|x\rangle|0\rangle^m|\psi\rangle$ the probability of finding the $(|x|+1)$st qubit in state $|1\rangle$ has probability at least $a_{|x|}$,

$x \notin L$ for any state $|\phi\rangle$ upon measuring the state $V_{|x|}|x\rangle|0\rangle^m|\phi\rangle$ the probability of finding the $(|x|+1)$st qubit in state $|1\rangle$ has probability at most $b_{|x|}$.
The language class QMA

The language $L$ belongs to the class QMA if for every input length $|x|$ there exists a quantum verifier $V_{|x|}$, and numbers $0 \leq b_{|x|} < a_{|x|} \leq 1$ satisfying $\frac{1}{a_{|x|} - b_{|x|}} = O\left(\text{poly}\left(|x|\right)\right)$, such that for all $x \in L$ there exists a witness $|\psi\rangle$ such that upon measuring the state $V_{|x|}|x\rangle|0\rangle^m|\psi\rangle$ the probability of finding the $(|x| + 1)$st qubit in state $|1\rangle$ has probability at least $a_{|x|}$,

$x \not\in L$ for any state $|\phi\rangle$ upon measuring the state $V_{|x|}|x\rangle|0\rangle^m|\phi\rangle$ the probability of finding the $(|x| + 1)$st qubit in state $|1\rangle$ has probability at most $b_{|x|}$.

Fast QMA amplification [Nagaj et al.’09]

We can modify the verifier circuit $V_{|x|}$ such that the acceptance probability thresholds become $a' := 1 - \varepsilon$ and $b' := \varepsilon$ using singular value transformation of degree $O\left(\frac{1}{\sqrt{a_{|x|}} - \sqrt{b_{|x|}}} \log \left(\frac{1}{\varepsilon}\right)\right)$. 
Fast QMA gap amplification [Marriott-Watrous’05] [Nagaj et al.’09]

The language class QMA

The language $L$ belongs to the class QMA if for every input length $|x|$ there exists a quantum verifier $V_{|x|}$, and numbers $0 \leq b_{|x|} < a_{|x|} \leq 1$ satisfying $\frac{1}{a_{|x|}-b_{|x|}} = O\left(\text{poly}\left(|x|\right)\right)$, such that for all $x \in L$ there exists a witness $|\psi\rangle$ such that upon measuring the state $V_{|x|}|x\rangle|0\rangle^m|\psi\rangle$ the probability of finding the $(|x|+1)$st qubit in state $|1\rangle$ has probability at least $a_{|x|}$, $x \notin L$ for any state $|\phi\rangle$ upon measuring the state $V_{|x|}|x\rangle|0\rangle^m|\phi\rangle$ the probability of finding the $(|x|+1)$st qubit in state $|1\rangle$ has probability at most $b_{|x|}$.

Fast QMA amplification [Nagaj et al.’09]

We can modify the verifier circuit $V_{|x|}$ such that the acceptance probability thresholds become $a' := 1 - \varepsilon$ and $b' := \varepsilon$ using singular value transformation of degree $O\left(\frac{1}{\sqrt{a_{|x|}} - \sqrt{b_{|x|}}} \log \left(\frac{1}{\varepsilon}\right)\right)$.

Observe that by the above definition

$\forall x \in L : \left\| \left( |x\rangle \otimes |1\rangle \otimes 1_{n+m-1} \right) V \left( |x\rangle \otimes |0\rangle^m \otimes I_n \right) \right\| \geq \sqrt{a_{|x|}}$,

$\forall x \notin L : \left\| \left( |x\rangle \otimes |1\rangle \otimes 1_{n+m-1} \right) V \left( |x\rangle \otimes |0\rangle^m \otimes I_n \right) \right\| \leq \sqrt{b_{|x|}}$. 
Singular vector transformation and projection

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

Amplitude amplification problem: Given $U$ such that

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

prepare $|\psi_{\text{good}}\rangle$. 

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

Generalization: Singular vector transformation

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

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

is a singular value decomposition.

Transform one copy of a quantum state $|\psi\rangle = \sum_{i} \alpha_i |\psi_i\rangle$ to $|\phi\rangle = \sum_{i} \alpha_i |\phi_i\rangle$.

If $\varsigma_i \geq \delta$ for all $\alpha_i$, we can $\epsilon$-apx. using QSVT with compl. $O(\frac{1}{\epsilon \log(\frac{1}{\epsilon})})$. 


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

Amplitude amplification problem: Given $U$ such that

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

prepare $|\psi_{\text{good}}\rangle$.

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

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

Amplitude amplification problem: Given $U$ such that

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

prepare $|\psi_{\text{good}}\rangle$.

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

**Generalization: Singular vector transformation**

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

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

is a singular value decomposition.
Singular vector transformation and projection

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

Amplitude amplification problem: Given $U$ such that

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

prepare $|\psi_{\text{good}}\rangle$.

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

Generalization: Singular vector transformation

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

$$A = \tilde{\Pi} U \Pi = \sum_{i=1}^{k} \varsigma_{i} |\phi_{i}\rangle \langle \psi_{i}|$$

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

$$|\psi\rangle = \sum_{i=1}^{k} \alpha_{i} |\psi_{i}\rangle \quad \text{to} \quad |\phi\rangle = \sum_{i=1}^{k} \alpha_{i} |\phi_{i}\rangle.$$
Singular vector transformation and projection

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

Amplitude amplification problem: Given $U$ such that

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

prepare $|\psi_{\text{good}}\rangle$.

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

Generalization: Singular vector transformation

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

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

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

$$|\psi\rangle = \sum_{i=1}^{k} \alpha_i |\psi_i\rangle$$

to

$$|\phi\rangle = \sum_{i=1}^{k} \alpha_i |\phi_i\rangle.$$

If $\varsigma_i \geq \delta$ for all $0 \neq \alpha_i$, we can $\varepsilon$-apx. using QSVT with compl. $O\left(\frac{1}{\delta} \log\left(\frac{1}{\varepsilon}\right)\right)$. 
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 $\varepsilon$ close to $e^{itH}$. Can be achieved with query complexity $O(t + \log(1/\varepsilon))$. Gate complexity is $O(a)$ times the above.

Proof sketch

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

Optimal complexity $\Theta(t + \log(1/\varepsilon) \log(e + \log(1/\varepsilon)/t))$ cf. density matrix exp. $\Theta(t^2/\varepsilon)$ Lloyd et al., Kimmel et al.
Suppose that $H$ is given as an $a$-qubit block-encoding, i.e., $U = \begin{bmatrix} H & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \end{bmatrix}$.

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

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

$$O(t + \log(1/\varepsilon)).$$

Gate complexity is $O(a)$ times the above.
Optimal block-Hamiltonian simulation

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

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

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

$$O(t + \log(1/\varepsilon)).$$

Gate complexity is $O(a)$ times the above.

Proof sketch

Approximate to $\varepsilon$-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., $U = \begin{bmatrix} H & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$.

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

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

$$O(t + \log(1/\varepsilon)).$$

Gate complexity is $O(a)$ times the above.

Proof sketch

Approximate to $\varepsilon$-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.]
Quantum speed-ups for distribution testing

The basic approach

$\text{Sample } i \sim p_i \text{ }$  

$\text{Estimate } \tilde{p}_i \text{ }$  

$\text{Output } f(\tilde{p}_i) \text{ }$  

E.g., for entropy output $-\log(\tilde{p}_i)$

$\text{Estimate } E[f(\tilde{p}_i)] \text{ by repeating the process}$

Quantum improvement: use amplitude estimation (Bravyi, Harrow and Hassidim – 2009)

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

Observation: a block encoding of $\sum_i \sqrt{p_i} |\tilde{\phi}_i\rangle \langle i|$ suffices and can be constructed!

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 $\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|$
Quantum speed-ups for distribution testing

The basic approach

- Sample $i \sim p_i$
Quantum speed-ups for distribution testing

The basic approach

- Sample $i \sim p_i$
- Estimate $\hat{p}_i$
Quantum speed-ups for distribution testing

The basic approach

- Sample $i \sim p_i$
- Estimate $\hat{p}_i$
- Output $f(\hat{p}_i)$

E.g., for entropy output $-\log(\hat{p}_i)$

Quantum improvement: use amplitude estimation (Bravyi, Harrow and Hassidim – 2009)

Suppose we can implement "quantum sampling":

$U_{p^i}: |0\rangle \mapsto \sum_i \sqrt{p_i} |\phi^i\rangle |i\rangle$

Observation: a block encoding of

$\sum_i \sqrt{p_i} |\tilde{\phi}^i\rangle \langle i|$ suffices and can be constructed!

The same technique works for density operators!

Purified access

$U_{\rho^i}: |0\rangle \mapsto \sum_i \sqrt{p_i} |\phi^i\rangle |\psi^i\rangle$, where $\rho = \sum_i p_i |\psi^i\rangle \langle \psi^i|$.
Quantum speed-ups for distribution testing

The basic approach

- Sample $i \sim p_i$
- Estimate $\tilde{p}_i$
- Output $f(\tilde{p}_i)$ E.g., for entropy output – $\log(\tilde{p}_i)$
Quantum speed-ups for distribution testing

The basic approach

- Sample $i \sim p_i$
- Estimate $\tilde{p}_i$
- Output $f(\tilde{p}_i)$ E.g., for entropy output – $\log(\tilde{p}_i)$
- Estimate $\mathbb{E}[f(\tilde{p}_i)]$ by repeating the process
Quantum speed-ups for distribution testing

The basic approach

- Sample $i \sim p_i$
- Estimate $\tilde{p}_i$
- Output $f(\tilde{p}_i)$ E.g., for entropy output $-\log(\tilde{p}_i)$
- Estimate $\mathbb{E}[f(\tilde{p}_i)]$ by repeating the process

Quantum improvement:

- Suppose we can implement "quantum sampling": $U_{p_i}: |0\rangle \mapsto \sum_i \sqrt{p_i} |\phi_i\rangle |i\rangle$
- Observation: a block encoding of $\sum_i \sqrt{p_i} |\tilde{\phi}_i\rangle$ suffices and can be constructed!
- 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 $\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|$
Quantum speed-ups for distribution testing

The basic approach

- Sample $i \sim p_i$
- Estimate $\tilde{p}_i$
- Output $f(\tilde{p}_i)$ E.g., for entropy output $-\log(\tilde{p}_i)$
- Estimate $\mathbb{E}[f(\tilde{p}_i)]$ by repeating the process

Quantum improvement: use amplitude estimation (Bravyi, Harrow and Hassidim – 2009)

Suppose we can implement "quantum sampling": $U_p : |0\rangle \mapsto \sum_i \sqrt{p_i} |\phi_i\rangle |i\rangle$
Quantum speed-ups for distribution testing

The basic approach

- Sample $i \sim p_i$
- Estimate $\tilde{p}_i$
- Output $f(\tilde{p}_i)$ E.g., for entropy output – $\log(\tilde{p}_i)$
- Estimate $\mathbb{E}[f(\tilde{p}_i)]$ by repeating the process

Quantum improvement: use amplitude estimation (Bravyi, Harrow and Hassidim – 2009)

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

Observation: a block encoding of $\sum_i \sqrt{p_i} |\tilde{\phi_i} \times i\rangle$ suffices and can be constructed!
Quantum speed-ups for distribution testing

The basic approach

- Sample $i \sim p_i$
- Estimate $\tilde{p}_i$
- Output $f(\tilde{p}_i)$ E.g., for entropy output $-\log(\tilde{p}_i)$
- Estimate $\mathbb{E}[f(\tilde{p}_i)]$ by repeating the process

Quantum improvement: use amplitude estimation (Bravyi, Harrow and Hassidim – 2009)

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

Observation: a block encoding of $\sum_i \sqrt{p_i} |\tilde{\phi}_i\rangle |i\rangle$ suffices and can be constructed!

The same technique works for density operators!
Quantum speed-ups for distribution testing

The basic approach

- Sample \( i \sim p_i \)
- Estimate \( \tilde{p}_i \)
- Output \( f(\tilde{p}_i) \) E.g., for entropy output – \( \log(\tilde{p}_i) \)
- Estimate \( E[f(\tilde{p}_i)] \) by repeating the process

Quantum improvement: use amplitude estimation (Bravyi, Harrow and Hassidim – 2009)

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

Observation: a block encoding of \( \sum_i \sqrt{p_i}|\tilde{\phi}_i\rangle\langle i| \) suffices and can be constructed!

The same technique works for density operators!

Purified access \( U_\rho : |0\rangle \mapsto \sum_i \sqrt{\rho_i}|\phi_i\rangle|\psi_i\rangle \), where \( \rho = \sum_i p_i|\psi_i\rangle\langle \psi_i| \)
Trick - skip estimating $p_i$

Operationally access and transform the probabilities:

$$U_p$$

Estimate the probability of measuring $|0\rangle$:

$$\sum_{i=1}^{n} p_i f(p_i) = E[f(p_i)]$$
Trick - skip estimating $p_i$

Operationally access and transform the probabilities:

$$U_p \Rightarrow U'_p := \text{diag}(\sqrt{p})$$

Apply the operation to a sample:

$$U'_p(p)|0\rangle \sum_{i=1}^{\text{N}} \sqrt{p_i}|i\rangle|\psi_i\rangle$$

Estimate the probability of measuring $|0\rangle$:

$$\sum_{i=1}^{\text{N}} p_i f(p_i) = E[f(p)]$$
Trick - skip estimating $p_i$

Operationally access and transform the probabilities:

\[
U_p \Rightarrow U'_p := \begin{bmatrix}
\text{diag}(\sqrt{p}) & \cdot \\
\cdot & \cdot
\end{bmatrix} \xrightarrow{\text{QSVT}} U'_{f(p)} := \begin{bmatrix}
\text{diag}(\sqrt{f(p)}) & \cdot \\
\cdot & \cdot
\end{bmatrix}
\]
Trick - skip estimating $p_i$

Operationally access and transform the probabilities:

$$U_p \Rightarrow U'_p := \begin{bmatrix} \text{diag}(\sqrt{p}) & \cdots \end{bmatrix} \xrightarrow{\text{QSVT}} U'_{f(p)} := \begin{bmatrix} \text{diag}(\sqrt{f(p)}) & \cdots \end{bmatrix}$$

Apply the operation to a sample:

$$U'_{f(p)} |0\rangle \sum_{i=1} \sqrt{p_i} |i\rangle |\psi_i\rangle = |0\rangle \sum_{i=1} \sqrt{p_i} \sqrt{f(p_i)} |i\rangle |\tilde{\psi}_i\rangle + |1\rangle \ldots$$

Estimate the probability of measuring $|0\rangle$:

$$\sum_{i=1} p_i f(p_i) = E[f(p)]$$
Trick - skip estimating $p_i$

Operationally access and transform the probabilities:

$$U_p \Rightarrow U_p' := \begin{bmatrix} \text{diag}(\sqrt{p}) & \cdots \\ \cdots & \cdots \end{bmatrix} \xrightarrow{\text{QSVT}} U_{f(p)}' := \begin{bmatrix} \text{diag}(\sqrt{f(p)}) & \cdots \\ \cdots & \cdots \end{bmatrix}$$

Apply the operation to a sample:

$$U_{f(p)}' |0\rangle \sum_{i=1} \sqrt{p_i} |i\rangle |\psi_i\rangle = |0\rangle \sum_{i=1} \sqrt{p_i} \sqrt{f(p_i)} |i\rangle |\tilde{\psi}_i\rangle + |1\rangle \ldots$$

Estimate the probability of measuring $|0\rangle$:

$$\sum_{i=1} p_i f(p_i) = \mathbb{E}[f(p)]$$
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 $f : [-1, 1] \rightarrow \mathbb{C}$, then implementing a block-encoding of $f(H)$ requires at least $\| \frac{df}{dx} \|_1$ 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

Let $I := [1/\kappa, 1/2]$ and let $f(x) := 1/\kappa x$, then $\left|\frac{df}{dx}\right|_{1/\kappa} = -\kappa$. Thus our implementation is optimal up to the $\log(1/\epsilon)$ factor.
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 $f : [-1, 1] \rightarrow \mathbb{C}$, then implementing a block-encoding of $f(H)$ requires at least $\| \frac{df}{dx} \|_1$ 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.
Lower bound on eigenvalue transformation

Suppose that $U$ is a block-encoding of a Hermitian matrix $H$ from a family of operators. Let $f: [-1, 1] \rightarrow \mathbb{C}$, then implementing a block-encoding of $f(H)$ requires at least $\| \frac{df}{dx} \|_1$ 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

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

Thus our implementation is optimal up to the $\log(1/\varepsilon)$ factor.
## Summarizing the various speed-ups

<table>
<thead>
<tr>
<th>Speed-up</th>
<th>Source of speed-up</th>
<th>Examples of algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential</td>
<td>Dimensionality of the Hilbert space</td>
<td>Hamiltonian simulation</td>
</tr>
<tr>
<td></td>
<td>Precise polynomial approximations</td>
<td>Improved HHL algorithm</td>
</tr>
<tr>
<td>Quadratic</td>
<td>Singular value = square root of probability</td>
<td>Grover search</td>
</tr>
<tr>
<td></td>
<td>Singular values are easier to distinguish</td>
<td>Amplitude estimation</td>
</tr>
<tr>
<td></td>
<td>Close-to-1 singular values are more flexible</td>
<td>Quantum walks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Summarizing the various speed-ups

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

### 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)
- Sample and gate efficient metrology, fractional queries
- ...
Summary of some applications of QSVT

$\sin(tx), \cos(tx)$:

Hamiltonian simulation

$\exp(-\beta x)$:

Gibbs sampling

$T_n(x)$:

$n = 13, 25$

Grover search

Ampl. ampl.

Quantum walks

$\approx \text{Heaviside}(x)$:

"Fixed-point" ampl. ampl.

Ground state prep.
Summary of some applications of QSVT

- **sin(tx), cos(tx):** Hamiltonian simulation
- **exp(−βx):** Gibbs sampling

\[
\sin(tx), \cos(tx): \quad \text{Exp}(−βx):\]

- Grover search
- "Fixed-point" ampl.
- Ground state prep.
Summary of some applications of QSVT

- $\sin(tx), \cos(tx)$: Hamiltonian simulation
- $\exp(-\beta x)$: Gibbs sampling
- $T_n(x)$: Grover search

Approximations:
- $\approx$ Heaviside
- "Fixed-point" amplification
- Ground state preparation

$n = 13, 25$
Summary of some applications of QSVT

\[ \sin(tx), \cos(tx) : \]

Hamiltonian simulation

\[ \exp(-\beta x) : \]

Gibbs sampling

\[ T_n(x) : \]

Grover search
Quantum walks

\[ \approx \text{Heaviside}(x) : \]

“Fixed-point” ampl. ampl.
Ground state prep.