Quantum element-wise transforms

Zane Rossi[1], Rahul Sarkar[2]

The University of Tokyo[1]
University of California, Berkeley[2]

Quantum element-wise transforms

or, QEWTs 

[2606.06456]

Zane Rossi[1], Rahul Sarkar[2]

The University of Tokyo[1]
University of California, Berkeley[2]

Quantum algorithms for numerical linear algebra

Quantum algorithms for numerical linear algebra

Numerical linear algebra undergirds the study of classical algorithms [Tre19, TB22]

Quantum algorithms for numerical linear algebra

Numerical linear algebra undergirds the study of classical algorithms [Tre19, TB22]

Matrix methods pervade signal processing, differential equations, statistics, machine learning, and the discrete analogues of huge swaths of functional analysis! [HJ85, HJ91]

Quantum algorithms for numerical linear algebra

Numerical linear algebra undergirds the study of classical algorithms [Tre19, TB22]

Matrix methods pervade signal processing, differential equations, statistics, machine learning, and the discrete analogues of huge swaths of functional analysis! [HJ85, HJ91]

Quantum algorithms are not unique in this [HHL09, CW12]; our understanding of most quantum algorithms is rooted in matrix methods [GSLW19, MRTC21]

Quantum algorithms for numerical linear algebra

Numerical linear algebra undergirds the study of classical algorithms [Tre19, TB22]

Matrix methods pervade signal processing, differential equations, statistics, machine learning, and the discrete analogues of huge swaths of functional analysis! [HJ85, HJ91]

Quantum algorithms are not unique in this [HHL09, CW12]; our understanding of most quantum algorithms is rooted in matrix methods [GSLW19, MRTC21]

The result — a growing, numerically-stable toolkit to interpret, optimize, and communicate quantum algorithms

The modern quantum algorithmic landscape

The modern quantum algorithmic landscape

Inversion, phase estimation, QR/LU/polar decomps, SVDs, exponentiation, linear combinations, diffeqs, matrix equations, contour integrals …

The modern quantum algorithmic landscape

Inversion, phase estimation, QR/LU/polar decomps, SVDs, exponentiation, linear combinations, diffeqs, matrix equations, contour integrals …

$$A \mapsto A^{-1}$$

$$\dot{u} = A(t)u + b(t)$$

$$A \mapsto e^{iAt}$$

$$e^{(A+iB)t} = \int f(k) e^{(kA+B)t}\, dk$$

$$(A, B) \mapsto f(A, B)$$

The modern quantum algorithmic landscape

Inversion, phase estimation, QR/LU/polar decomps, SVDs, exponentiation, linear combinations, diffeqs, matrix equations, contour integrals …

$$A \mapsto A^{-1}$$

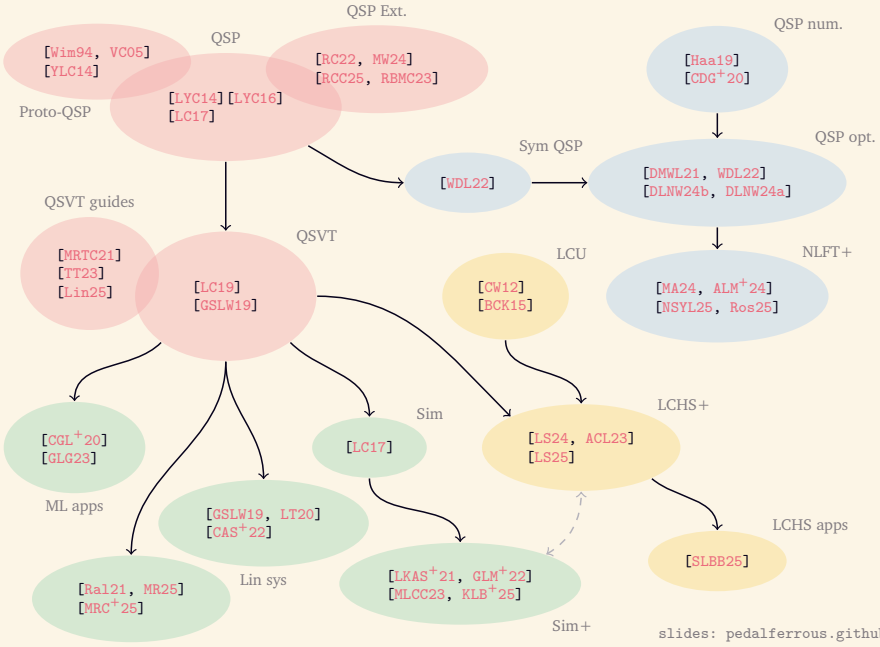
$$\dot{u} = A(t)u + b(t)$$

$$A \mapsto e^{iAt}$$

$$e^{(A+iB)t} = \int f(k) e^{(kA+B)t}\, dk$$

$$(A, B) \mapsto f(A, B)$$

These methods are incredibly flexible, and allow importation of/comparison with diverse classical techniques!



slides: pedalferrous.github.io

Block encoding basics

Block encoding basics

We say U is an (α, a, ε) block encoding (BE) of an n-qubit operator A if

Block encoding basics

We say U is an (α, a, ε) block encoding (BE) of an n-qubit operator A if

$$\left\| (\langle 0^a | \otimes I_n) U (|0^a\rangle \otimes I_n) - A/\alpha \right\| \leq \varepsilon$$

Block encoding basics

We say U is an (α, a, ε) block encoding (BE) of an n-qubit operator A if

$$U = \begin{bmatrix} A/\alpha & * \\ * & * \end{bmatrix}$$

Block encoding basics

We say U is an (α, a, ε) block encoding (BE) of an n-qubit operator A if

$$U = \begin{bmatrix} A/\alpha & * \\ * & * \end{bmatrix}$$

Captures subsystem dynamics; if aux space begins and ends in $|0^a\rangle$, then A is applied to n-qubit input $|\psi\rangle$

Block encoding basics

We say U is an (α, a, ε) block encoding (BE) of an n-qubit operator A if

$$U = \begin{bmatrix} A/\alpha & * \\ * & * \end{bmatrix}$$

Captures subsystem dynamics; if aux space begins and ends in $|0^a\rangle$, then A is applied to n-qubit input $|\psi\rangle$

Algorithms like QSVT[1] [GSLW19] and LCU[2] [CW12] allow spectral maps and simple matrix arithmetic; e.g., $A \mapsto f(A)$, or $(A, B) \mapsto \gamma A + \delta B$

[1] Quantum singular value transformation
[2] Linear combination of unitaries

Block encoding basics

We say U is an (α, a, ε) block encoding (BE) of an n-qubit operator A if

$$U = \begin{bmatrix} A/\alpha & * \\ * & * \end{bmatrix}$$

Captures subsystem dynamics; if aux space begins and ends in $|0^a\rangle$, then A is applied to n-qubit input $|\psi\rangle$

Algorithms like QSVT[1] [GSLW19] and LCU[2] [CW12] allow spectral maps and simple matrix arithmetic; e.g., $A \mapsto f(A)$, or $(A, B) \mapsto \gamma A + \delta B$

Important questions include BE cost, query complexity, subnormalization/error growth, and space use!

[1] Quantum singular value transformation
[2] Linear combination of unitaries

Block encoding basics

We say U is an (α, a, ε) block encoding (BE) of an n-qubit operator A if

$$U = \begin{bmatrix} A/\alpha & * \\ * & * \end{bmatrix}$$

Captures subsystem dynamics; if aux space begins and ends in $|0^a\rangle$, then A is applied to n-qubit input $|\psi\rangle$

Algorithms like QSVT[1] [GSLW19] and LCU[2] [CW12] allow spectral maps and simple matrix arithmetic; e.g., $A \mapsto f(A)$, or $(A, B) \mapsto \gamma A + \delta B$

Important questions include BE cost, query complexity, subnormalization/error growth, and space use!

For BE algorithms, these relate to concrete questions in functional analysis and numerical linear algebra!

A note on matrix algebras

A note on matrix algebras

This work concerns the efficient realization of various matrix manipulations by quantum algorithms

A note on matrix algebras

Square matrices over $\mathbb{C}$ admit associative algebra, i.e., addition, multiplication, scalar multiplication

$$C_{ij} = \sum_k A_{ik} B_{kj} \leftrightarrow C = AB$$

A note on matrix algebras

Functions of matrices are often defined spectrally, and algorithms like QSVT [GSLW19, MRTC21] act on the singular values, e.g., if $M = U\Sigma V^\dagger$

$$f^{(\mathrm{SV})}(M) \equiv U f(\Sigma) V^\dagger = U \begin{bmatrix} f(\xi_0) & & \\ & \ddots & \\ & & f(\xi_{n-1}) \end{bmatrix} V^\dagger, \quad \Sigma = \mathrm{diag}(\xi_0, \xi_1, \ldots, \xi_{n-1}),$$

A note on matrix algebras

However, there exist other associative algebras for matrices over fields, including the element-wise product[1]

[1] Variously referred to as the Schur, Hadamard, or entry-wise product.

A note on matrix algebras

However, there exist other associative algebras for matrices over fields, including the element-wise product[1]

This product appears ubiquitously in classical signal processing, machine learning, and statistics, and has no simple spectral interpretation! [HJ85, HJ91, CWP+25, Sty73] E.g., consider $f^{\circ}(A)$

[1] Variously referred to as the Schur, Hadamard, or entry-wise product.

A note on matrix algebras

However, there exist other associative algebras for matrices over fields, including the element-wise product[1]

This product appears ubiquitously in classical signal processing, machine learning, and statistics, and has no simple spectral interpretation! [HJ85, HJ91, CWP+25, Sty73] E.g., consider $f^{\circ}(A)$

$$\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0,m-1} \\ a_{10} & a_{11} & & \vdots \\ \vdots & & \ddots & \\ a_{n-1,0} & \cdots & & a_{n-1,m-1} \end{bmatrix} \mapsto \begin{bmatrix} f(a_{00}) & f(a_{01}) & \cdots & f(a_{0,m-1}) \\ f(a_{10}) & f(a_{11}) & & \vdots \\ \vdots & & \ddots & \\ f(a_{n-1,0}) & \cdots & & f(a_{n-1,m-1}) \end{bmatrix}.$$

[1] Variously referred to as the Schur, Hadamard, or entry-wise product.

A note on matrix algebras

However, there exist other associative algebras for matrices over fields, including the element-wise product[1]

This product appears ubiquitously in classical signal processing, machine learning, and statistics, and has no simple spectral interpretation! [HJ85, HJ91, CWP+25, Sty73] E.g., consider $f^{\circ}(A)$

$$\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0,m-1} \\ a_{10} & a_{11} & & \vdots \\ \vdots & & \ddots & \\ a_{n-1,0} & \cdots & & a_{n-1,m-1} \end{bmatrix} \mapsto \begin{bmatrix} f(a_{00}) & f(a_{01}) & \cdots & f(a_{0,m-1}) \\ f(a_{10}) & f(a_{11}) & & \vdots \\ \vdots & & \ddots & \\ f(a_{n-1,0}) & \cdots & & f(a_{n-1,m-1}) \end{bmatrix}.$$

Despite this product's ubiquity, quantum algorithms for element-wise manipulations are only cursorily understood [GYC+25] …

[1] Variously referred to as the Schur, Hadamard, or entry-wise product.

A note on matrix algebras

$\otimes$	`\otimes`	(Kronecker/tensor product),
$\circ$	`\circ`	(Element-wise product),
$\boxtimes$	`\boxtimes`	(Tracy–Singh product),
$\odot$	`\odot`	(Khatri-Rao product),
$*$	`\ast`	(Convolution).

Overview

Overview

Introduce informal main result and related prior work

Overview

Introduce informal main result and related prior work

Build core subroutines: swap copy and weaving lemma

Overview

Introduce informal main result and related prior work

Build core subroutines: swap copy and weaving lemma

Recursively construct full QEWT circuit

Overview

Introduce informal main result and related prior work

Build core subroutines: swap copy and weaving lemma

Recursively construct full QEWT circuit

Discuss properties, errors in prior work, & applications

Overview

Introduce informal main result and related prior work

Build core subroutines: swap copy and weaving lemma

Recursively construct full QEWT circuit

Discuss properties, errors in prior work, & applications

Give outlook, extensions, and high-level takeaways

Main result

Main result

Quantum element-wise transform; *QEWT*[1]

Let f be a polynomial of degree d satisfying $|f| \leq 1$ on $[-1,1]$, with coefficients c_k. Then, given access to controlled U, where U is an (α, a, ε) block encoding of an n-qubit operator A, there exists a quantum circuit preparing a (β, b, δ) block encoding of $f^{\circ}(A/\alpha)$ using d copies of controlled U, where (β, b, δ) satisfy

$$\beta = \sum_{k=1}^{d} |c_k|, \quad b = \mathcal{O}(a + n \log d), \quad \delta = \sum_{k=1}^{d} |c_k| \left[(1+\varepsilon/\alpha)^k - 1 \right].$$

[1]Pronounced *newt*

Main result

Quantum element-wise transform; *QEWT*[1]

Let f be a polynomial of degree d satisfying $|f| \leq 1$ on $[-1,1]$, with coefficients c_k. Then, given access to controlled U, where U is an (α, a, ε) block encoding of an n-qubit operator A, there exists a quantum circuit preparing a (β, b, δ) block encoding of $f^{\circ}(A/\alpha)$ using d copies of controlled U, where (β, b, δ) satisfy

$$\beta = \sum_{k=1}^{d} |c_k|, \quad b = \mathcal{O}(a + n \log d), \quad \delta = \sum_{k=1}^{d} |c_k| \left[(1+\varepsilon/\alpha)^k - 1 \right].$$

[1]Pronounced *newt*

Main result

Quantum element-wise transform; *QEWT*[1]

Let f be a polynomial of degree d satisfying $|f| \leq 1$ on $[-1,1]$, with coefficients c_k. Then, given access to controlled U, where U is an (α, a, ε) block encoding of an n-qubit operator A, there exists a quantum circuit preparing a (β, b, δ) block encoding of $f^{\circ}(A/\alpha)$ using d copies of controlled U, where (β, b, δ) satisfy

$$\beta = \sum_{k=1}^{d} |c_k|, \quad b = \mathcal{O}(a + n \log d), \quad \delta = \sum_{k=1}^{d} |c_k| \left[(1+\varepsilon/\alpha)^k - 1 \right].$$

[1]Pronounced *newt*

Main result

Quantum element-wise transform; *QEWT*[1]

Let f be a polynomial of degree d satisfying $|f| \leq 1$ on $[-1,1]$, with coefficients c_k. Then, given access to controlled U, where U is an (α, a, ε) block encoding of an n-qubit operator A, there exists a quantum circuit preparing a (β, b, δ) block encoding of $f^{\circ}(A/\alpha)$ using d copies of controlled U, where (β, b, δ) satisfy

$$\beta = \sum_{k=1}^{d} |c_k|, \quad b = \mathcal{O}(a + n \log d), \quad \delta = \sum_{k=1}^{d} |c_k| \left[(1+\varepsilon/\alpha)^k - 1 \right].$$

[1]Pronounced *newt*

Main result

Quantum element-wise transform; *QEWT*[1]

Let f be a polynomial of degree d satisfying $|f| \leq 1$ on $[-1,1]$, with coefficients c_k. Then, given access to controlled U, where U is an (α, a, ε) block encoding of an n-qubit operator A, there exists a quantum circuit preparing a (β, b, δ) block encoding of $f^{\circ}(A/\alpha)$ using d copies of controlled U, where (β, b, δ) satisfy

$$\beta = \sum_{k=1}^{d} |c_k|, \quad b = \mathcal{O}(a + n \log d), \quad \delta = \sum_{k=1}^{d} |c_k| \left[(1+\varepsilon/\alpha)^k - 1 \right].$$

For commonly applied functions & precisions this is easily 15–20× space saving, and offers a new, cheap way to induce non-linear transformations of high-dim data

[1] Pronounced *newt*

Main result

Quantum element-wise transform; *QEWT*[1]

Let f be a polynomial of degree d satisfying $|f| \leq 1$ on $[-1,1]$, with coefficients c_k. Then, given access to controlled U, where U is an (α, a, ε) block encoding of an n-qubit operator A, there exists a quantum circuit preparing a (β, b, δ) block encoding of $f^\circ(A/\alpha)$ using d copies of controlled U, where (β, b, δ) satisfy

$$\beta = \sum_{k=1}^{d} |c_k|, \quad b = \mathcal{O}(a + n \log d), \quad \delta = \sum_{k=1}^{d} |c_k| \left[(1 + \varepsilon/\alpha)^k - 1 \right].$$

For commonly applied functions & precisions this is easily 15–20× space saving, and offers a new, cheap way to induce non-linear transformations of high-dim data

[1] Pronounced *newt*

Prior work

Prior work

Considers non-linear transformations of amplitudes [HCSS23, RR23, GMF24], as well as iterated element-wise products [GYC+25, RHG+25]

Prior work

Considers non-linear transformations of amplitudes [HCSS23, RR23, GMF24], as well as iterated element-wise products [GYC+25, RHG+25]

Far less is known/optimized for QEWTs,[1] and we identify an error in the linear combination of unitaries (LCU) based construction of [GYC+25]

[1] Space and query complexity in degree d, qubit num. n, block encoding aux. space a, and error ε

Prior work

Considers non-linear transformations of amplitudes [HCSS23, RR23, GMF24], as well as iterated element-wise products [GYC+25, RHG+25]

Far less is known/optimized for QEWTs,[1] and we identify an error in the linear combination of unitaries (LCU) based construction of [GYC+25]

Method	Query comp.	Aux. space
Prior work [GYC+25]	$\mathcal{O}(d)$	$\tilde{\mathcal{O}}(ad+nd)$
Main result (Thm. II.3)	$\mathcal{O}(d)$	$\mathcal{O}(a+n(\log d))$
LCU variant (Thm. III.4)	$\mathcal{O}(d)$	$\mathcal{O}(a+n(\log d))$
Approx. var. (Cor. III.5.2)	$\mathcal{O}(d^{\log(\log(d/\varepsilon))})$	$\mathcal{O}(a+n\log(d^{\log(\log(d/\varepsilon))}))$
Half-comp. (Thm. C.1)	$\mathcal{O}(d)$	$\tilde{\mathcal{O}}(a+nd)$

[1] Space and query complexity in degree d, qubit num. n, block encoding aux. space a, and error ε

Prior work

Considers non-linear transformations of amplitudes [HCSS23, RR23, GMF24], as well as iterated element-wise products [GYC+25, RHG+25]

Far less is known/optimized for QEWTs,[1] and we identify an error in the linear combination of unitaries (LCU) based construction of [GYC+25]

Method	Query comp.	Aux. space
Prior work [GYC+25]	$\mathcal{O}(d)$	$\tilde{\mathcal{O}}(ad+nd)$
Main result (Thm. II.3)	$\mathcal{O}(d)$	$\mathcal{O}(a+n(\log d))$
LCU variant (Thm. III.4)	$\mathcal{O}(d)$	$\mathcal{O}(a+n(\log d))$
Approx. var. (Cor. III.5.2)	$\mathcal{O}(d^{\log(\log(d/\varepsilon))})$	$\mathcal{O}(a+n\log(d^{\log(\log(d/\varepsilon))}))$
Half-comp. (Thm. C.1)	$\mathcal{O}(d)$	$\tilde{\mathcal{O}}(a+nd)$

[1] Space and query complexity in degree d, qubit num. n, block encoding aux. space a, and error ε

Hardness, related methods, and errata

Hardness, related methods, and errata

For diagonal block encodings, element-wise product and matrix product coincide

Hardness, related methods, and errata

For diagonal block encodings, element-wise product and matrix product coincide

Cheap methods to convert state preparation unitaries to diagonal block encodings inform nonlinear amplitude transformations [HCSS23, RR23, GMF24]

Hardness, related methods, and errata

For diagonal block encodings, element-wise product and matrix product coincide

Cheap methods to convert state preparation unitaries to diagonal block encodings inform nonlinear amplitude transformations [HCSS23, RR23, GMF24]

$$\begin{bmatrix} | & | & | \\ |\psi\rangle & * & * \\ | & | & | \end{bmatrix} \to \begin{bmatrix} \psi_0 & & \\ & \psi_1 & \\ & & \ddots \end{bmatrix} \to \begin{bmatrix} f(\psi_0) & & \\ & f(\psi_1) & \\ & & \ddots \end{bmatrix} \to \begin{bmatrix} | & | & | \\ |f(\psi)\rangle & * & * \\ | & | & | \end{bmatrix}$$

Hardness, related methods, and errata

For diagonal block encodings, element-wise product and matrix product coincide

Cheap methods to convert state preparation unitaries to diagonal block encodings inform nonlinear amplitude transformations [HCSS23, RR23, GMF24]

However, $\leftrightarrow$ vectorization comes with unacceptable dimension-dependent subnormalization, and should be avoided!

Hardness, related methods, and errata

For diagonal block encodings, element-wise product and matrix product coincide

Cheap methods to convert state preparation unitaries to diagonal block encodings inform nonlinear amplitude transformations [HCSS23, RR23, GMF24]

However, $\leftrightarrow$ vectorization comes with unacceptable dimension-dependent subnormalization, and should be avoided!

Moreover, applying linear combination of unitaries to non-matrix products requires care! [GYC+25]

Hardness, related methods, and errata

For diagonal block encodings, element-wise product and matrix product coincide

Cheap methods to convert state preparation unitaries to diagonal block encodings inform nonlinear amplitude transformations [HCSS23, RR23, GMF24]

However, $\leftrightarrow$ vectorization comes with unacceptable dimension-dependent subnormalization, and should be avoided!

Moreover, applying linear combination of unitaries to non-matrix products requires care! [GYC+25]

$$A \circ I = \mathrm{diag}(A) \neq A, \qquad (A^{\circ 2}) \times (A^{\circ 2}) \neq A^{\circ 4}.$$

Hardness, related methods, and errata

For diagonal block encodings, element-wise product and matrix product coincide

Cheap methods to convert state preparation unitaries to diagonal block encodings inform nonlinear amplitude transformations [HCSS23, RR23, GMF24]

However, $\leftrightarrow$ vectorization comes with unacceptable dimension-dependent subnormalization, and should be avoided!

Moreover, applying linear combination of unitaries to non-matrix products requires care! [GYC+25]

$$A \circ I = \mathrm{diag}(A) \neq A, \qquad (A^{\circ 2}) \times (A^{\circ 2}) \neq A^{\circ 4}.$$

All of this establishes query lower bounds, and space/success probability upper bounds, but leaves huge room for improvement!

A minimal example

A minimal example

Note — $A \circ B$ is *principal submatrix*[1] of $A \otimes B$

Prev methods [GYC+25] permute rows/cols of $A^{\otimes d}$ — order nd space!

[1]Common identity, but for introduction see [HJ85, HJ91]

A minimal example

I.e., if we have block encodings of A, B, acting on disjoint registers …

$$U_{A\otimes I} = \left[\begin{array}{cccc|c} a_0 & & a_1 & & \\ & a_0 & & a_1 & * \\ a_2 & & a_3 & & \\ & a_2 & & a_3 & \\ \hline & * & & & * \end{array}\right], \qquad U_{I\otimes B} = \left[\begin{array}{cccc|c} b_0 & b_1 & & & \\ b_2 & b_3 & & & * \\ & & b_0 & b_1 & \\ & & b_2 & b_3 & \\ \hline & * & & & * \end{array}\right],$$

A minimal example

…then a row and column permutation relocates $A \circ B$ to the top-left

$$U_{A\otimes B} = \left[\begin{array}{cccc|c} a_0b_0 & * & * & a_1b_1 & \\ * & * & * & * & \\ * & * & * & * & * \\ a_2b_2 & * & * & a_3b_3 & \\ \hline & * & & & * \end{array}\right] \longrightarrow U_{P(A\otimes B)P^\dagger} = \left[\begin{array}{cccc|c} a_0b_0 & a_1b_1 & * & * & \\ a_2b_2 & a_3b_3 & * & * & \\ * & * & * & * & * \\ * & * & * & * & \\ \hline & * & & & * \end{array}\right].$$

A minimal example

If we then mask unwanted elements by left/right matrix mult …

$$V \equiv \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & \\ & & & 0 \end{bmatrix} \Longrightarrow V[P(A \otimes B)P^{\dagger}]V = \begin{bmatrix} a_0 b_0 & a_1 b_1 & & \\ a_2 b_2 & a_3 b_3 & & \\ & & & \end{bmatrix},$$

A minimal example

…and find a way to space-efficiently copy the top-left block, then we can return to input block encoding form $(I \otimes *)$

$$\begin{bmatrix} a_0b_0 & a_1b_1 & & \\ a_2b_2 & a_3b_3 & & \\ & & * & * \\ & & * & * \end{bmatrix} \rightarrow \begin{bmatrix} a_0b_0 & a_1b_1 & & \\ a_2b_2 & a_3b_3 & & \\ & & a_0b_0 & a_1b_1 \\ & & a_2b_2 & a_3b_3 \end{bmatrix} = I \otimes (A \circ B).$$

A minimal example

caveat lector!

We have to control gate/query complexity, success prob while making sure matrix multiplication and currently unspecified *copy* operation together require only $\log d$ additional space

A minimal example

caveat lector!

We have to control gate/query complexity, success prob while making sure matrix multiplication and currently unspecified *copy* operation together require only $\log d$ additional space

A minimal example

caveat lector!

We have to control gate/query complexity, success prob while making sure matrix multiplication and currently unspecified *copy* operation together require only $\log d$ additional space

Up next — Space- and query-efficient techniques for manipulating block encodings with block structure, which can themselves be space-efficiently recursively composed …

A minimal example

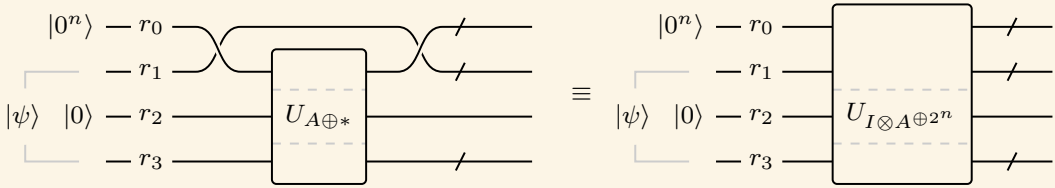
caveat lector!

We have to control gate/query complexity, success prob while making sure matrix multiplication and currently unspecified *copy* operation together require only $\log d$ additional space

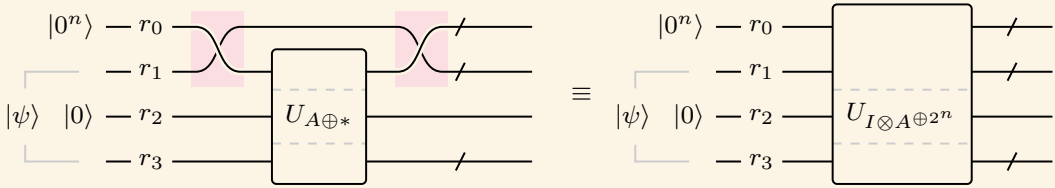
Up next — Space- and query-efficient techniques for manipulating block encodings with block structure, which can themselves be space-efficiently recursively composed …

Building the swap copy

Building the swap copy

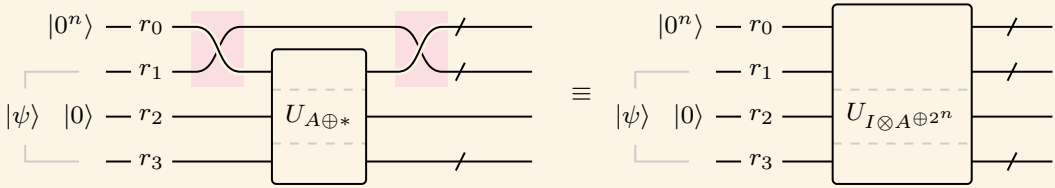


Building the swap copy



A type of *decontrollization*; i.e., decouple action of block encoding from state in r_1

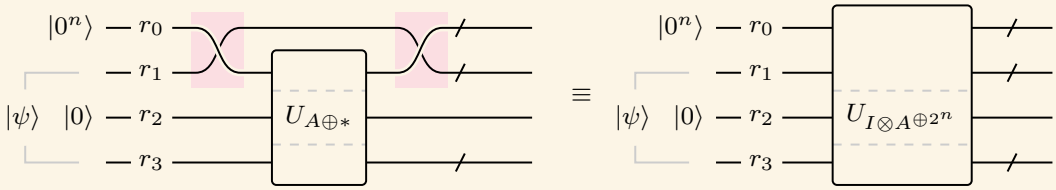
Building the swap copy



A type of *decontrollization*; i.e., decouple action of block encoding from state in r_1

Note 1 — the catalytic use of r_0 depends *only* on r_2 returning to $|0\rangle$

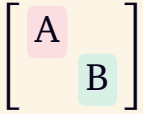
Building the swap copy

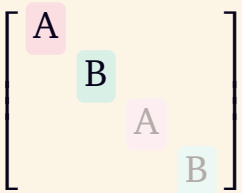


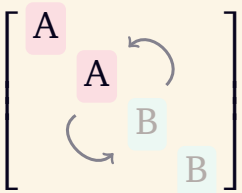
A type of *decontrollization*; i.e., decouple action of block encoding from state in r_1

Note 1 — the catalytic use of r_0 depends *only* on r_2 returning to $|0\rangle$

Note 2 — blocks of BEs are *not* a 'resource'; basis dependence means instantiating space instantiates blocks, which can be permuted around!


A
B


A
B
A
B


A
A
B
B

$$\begin{bmatrix} A & \\ & A \end{bmatrix}$$

On block encodings with block structure

On block encodings with block structure

Let $W(*_1, *_2)$ a function that takes and returns operators. Let S a set of *pairs of operators* and R a set of *quantum states* s.t. for all $(V_1, V_2) \in S$, all $|r\rangle \in R$:

On block encodings with block structure

Let $W(*_1, *_2)$ a function that takes and returns operators. Let S a set of *pairs of operators* and R a set of *quantum states* s.t. for all $(V_1, V_2) \in S$, all $|r\rangle \in R$:

$$W(V_1, V_2) = |r\rangle\langle r|_{r_0} \otimes U(V_1, V_2)_{r_1} + \sum_{r', r'' \neq r} |r'\rangle\langle r''|_{r_0} \otimes [*].$$

On block encodings with block structure

Let $W(*_1, *_2)$ a function that takes and returns operators. Let S a set of *pairs of operators* and R a set of *quantum states* s.t. for all $(V_1, V_2) \in S$, all $|r\rangle \in R$:

$$W(V_1, V_2) = |r\rangle\langle r|_{r_0} \otimes U(V_1, V_2)_{r_1} + \sum_{r', r'' \neq r} |r'\rangle\langle r''|_{r_0} \otimes [*].$$

Then we call W an (S, R)-blocked operator function

On block encodings with block structure

Let $W(*_1, *_2)$ a function that takes and returns operators. Let S a set of *pairs of operators* and R a set of *quantum states* s.t. for all $(V_1, V_2) \in S$, all $|r\rangle \in R$:

$$W(V_1, V_2) = |r\rangle\langle r|_{r_0} \otimes U(V_1, V_2)_{r_1} + \sum_{r', r'' \neq r} |r'\rangle\langle r''|_{r_0} \otimes [*].$$

Then we call W an (S, R)-blocked operator function

Moral — W, together with a copy[1] of $|r\rangle$, can simulate U; compositions of U_k can be realized by compositions of W_k using copies of $|r\rangle$

[1] Here freed after the use of W!

On block encodings with block structure

Let $W(*_1, *_2)$ a function that takes and returns operators. Let S a set of *pairs of operators* and R a set of *quantum states* s.t. for all $(V_1, V_2) \in S$, all $|r\rangle \in R$:

$$W(V_1, V_2) = |r\rangle\langle r|_{r_0} \otimes U(V_1, V_2)_{r_1} + \sum_{r', r'' \neq r} |r'\rangle\langle r''|_{r_0} \otimes [*].$$

Then we call W an (S, R)-blocked operator function

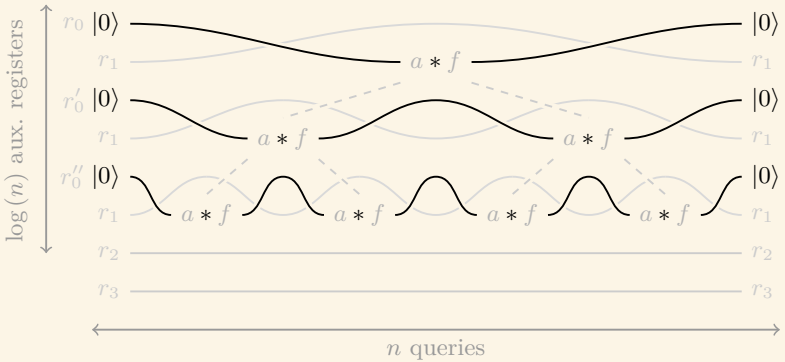
Moral — W, together with a copy[1] of $|r\rangle$, can simulate U; compositions of U_k can be realized by compositions of W_k using copies of $|r\rangle$

Weaving lemma connects $|r\rangle$-complexity to depth of function call tree

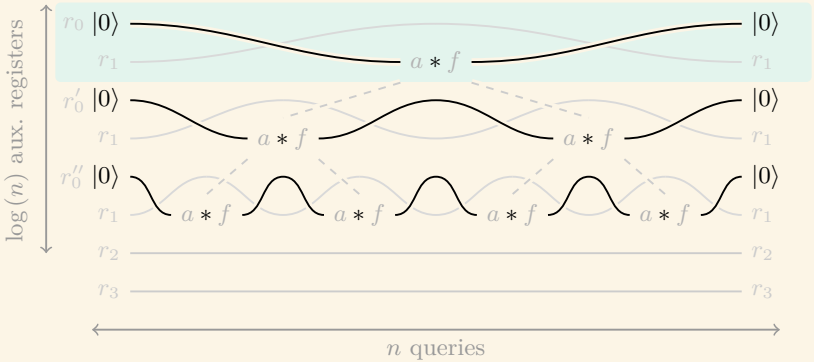
[1] Here freed after the use of W!

Nesting swap copy — the weaving lemma

Nesting swap copy — the weaving lemma



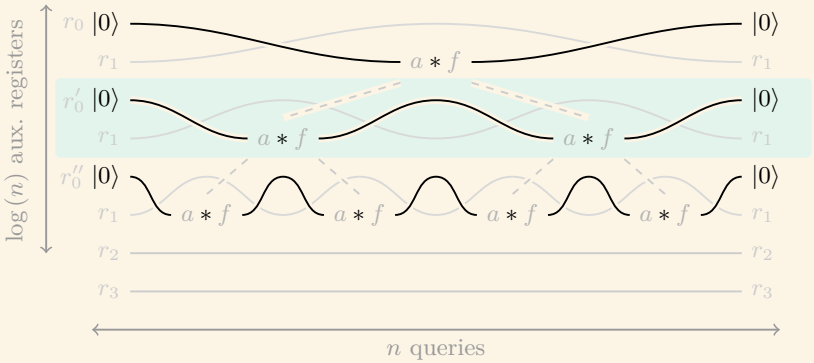
Nesting swap copy — the weaving lemma



Our *swap copy*-based element-wise product catalytically consumes[1] state in r_0, and can call subroutines which themselves require registers

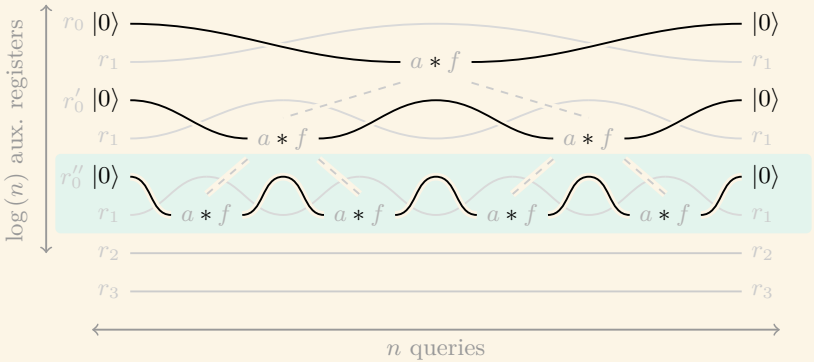
[1]Here denoted *allocates* and *frees*.

Nesting swap copy — the weaving lemma



At each level $r_0', r_0'', \ldots$ we will *track* successful BE applications to ensure catalysis — auxiliary space then scales with tree depth!

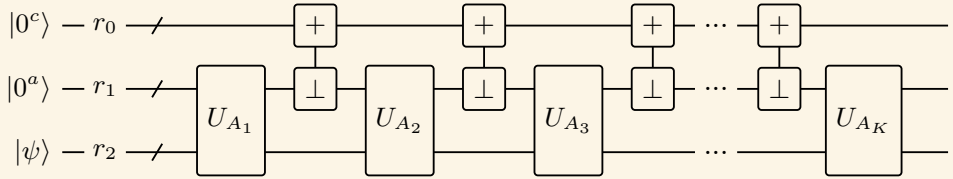
Nesting swap copy — the weaving lemma



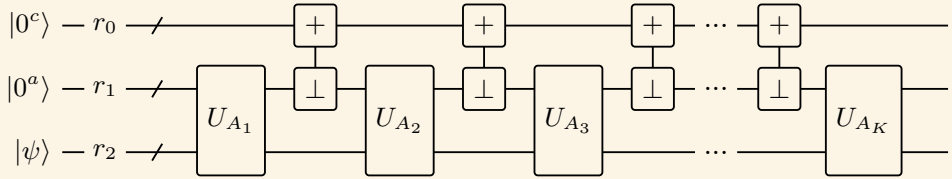
Space-efficient tracking of successful BE application is handled by *compression gadget* techniques [LW19] …

The compression gadget [LW19]

The compression gadget [LW19]

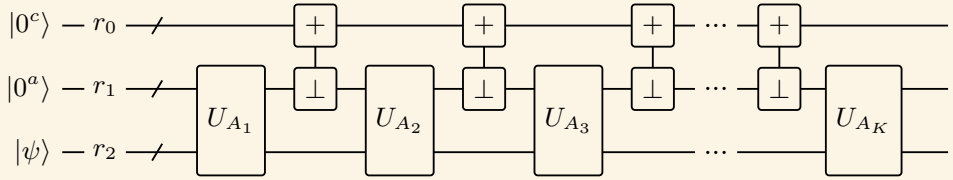


The compression gadget [LW19]



Given BEs of A_j, space-efficiently build BEs of products among A_j

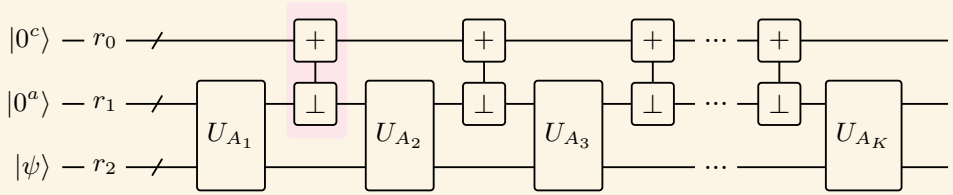
The compression gadget [LW19]



Given BEs of A_j, space-efficiently build BEs of products among A_j

Insight — controlled additions to *compression register*; lose information about which BE failed, but retain information that *at least one did*

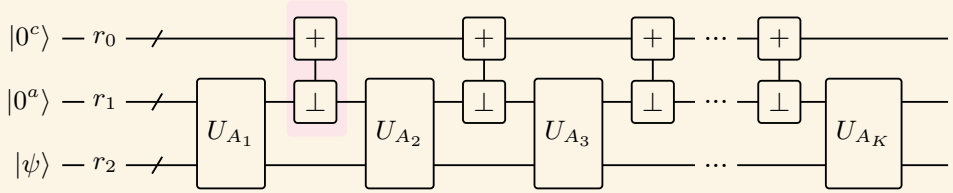
The compression gadget [LW19]



Given BEs of A_j, space-efficiently build BEs of products among A_j

Insight — controlled additions to *compression register*; lose information about which BE failed, but retain information that *at least one did*

The compression gadget [LW19]

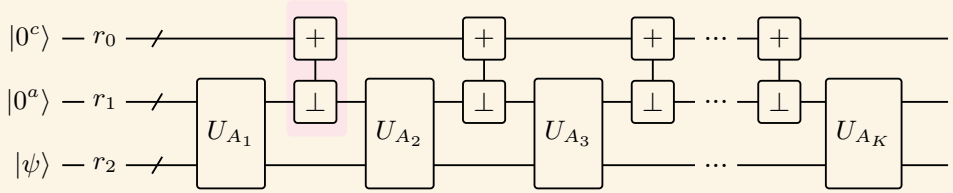


Given BEs of A_j, space-efficiently build BEs of products among A_j

Insight — controlled additions to *compression register*; lose information about which BE failed, but retain information that *at least one did*

Linear to logarithmic auxiliary space in product degree!

The compression gadget [LW19]



Given BEs of A_j, space-efficiently build BEs of products among A_j

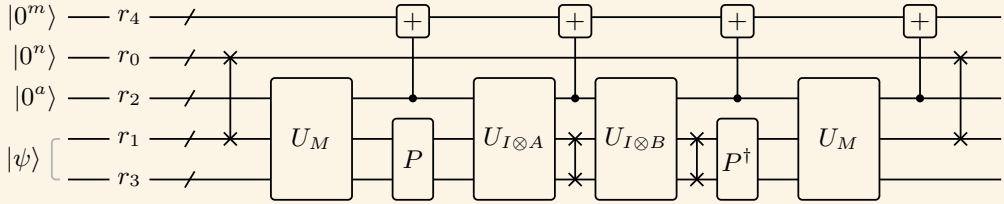
Insight — controlled additions to *compression register*; lose information about which BE failed, but retain information that *at least one did*

$$(\langle 0|_{ac} \otimes I_s) V (|0\rangle_{ac} \otimes I_s) = |0\rangle\langle 0|_b \otimes I_s + \sum_{k=0}^{K-1} |k\rangle\langle k|_b \otimes \Big[\prod_{j=0}^{k} A_j\Big]_s,$$

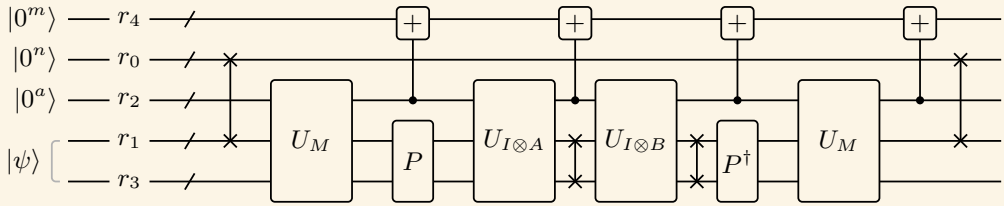
$$W = I_s \oplus \bigoplus_{k=0}^{K-1} \Big[\prod_{j=0}^{k} A_j\Big]_s,$$

Element-wise product — base case

Element-wise product — base case



Element-wise product — base case



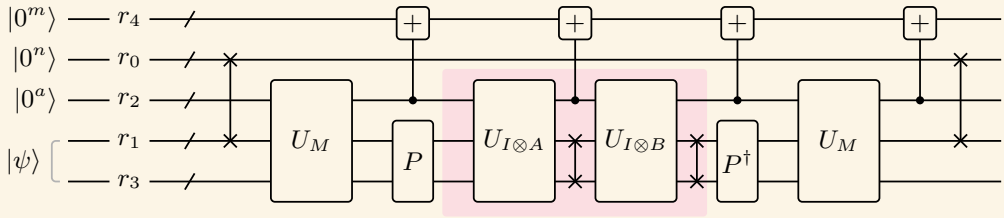
Standard (compressed) method for *matrix product*[1] between M, $P(A \otimes B)P^\dagger$, $M^\dagger$

Followed by *swap copy*, yielding block encoding (BE) of $I \otimes (A \circ B)$

Register r_4 records successful BE application by controlled-additions from r_2. Here r_1 and r_3 store product, and $r_0 \leftrightarrow r_1$ is *swap copy*

[1]This is a block diagonal block encoding containing $A \circ B$!

Element-wise product — base case



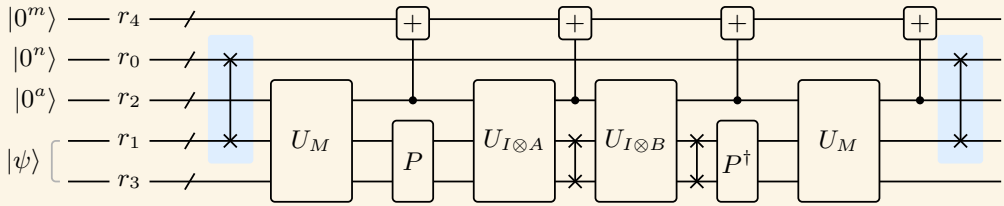
Standard (compressed) method for *matrix product*[1] between M, $P(A \otimes B)P^\dagger$, $M^\dagger$

Followed by *swap copy*, yielding block encoding (BE) of $I \otimes (A \circ B)$

Register r_4 records successful BE application by controlled-additions from r_2. Here r_1 and r_3 store product, and $r_0 \leftrightarrow r_1$ is *swap copy*

[1]This is a block diagonal block encoding containing $A \circ B$!

Element-wise product — base case



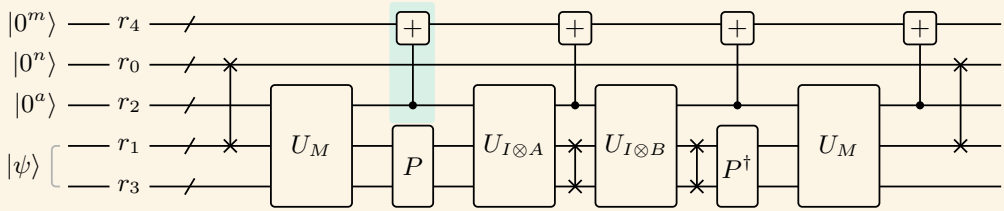
Standard (compressed) method for *matrix product*[1] between M, $P(A \otimes B)P^\dagger$, $M^\dagger$

Followed by *swap copy*, yielding block encoding (BE) of $I \otimes (A \circ B)$

Register r_4 records successful BE application by controlled-additions from r_2. Here r_1 and r_3 store product, and $r_0 \leftrightarrow r_1$ is *swap copy*

[1]This is a block diagonal block encoding containing $A \circ B$!

Element-wise product — base case



Standard (compressed) method for *matrix product*[1] between M, $P(A \otimes B)P^\dagger$, $M^\dagger$

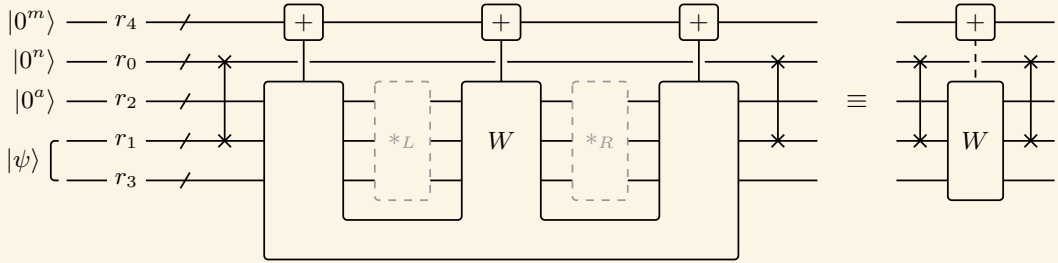
Followed by *swap copy*, yielding block encoding (BE) of $I \otimes (A \circ B)$

Register r_4 records successful BE application by controlled-additions from r_2. Here r_1 and r_3 store product, and $r_0 \leftrightarrow r_1$ is *swap copy*

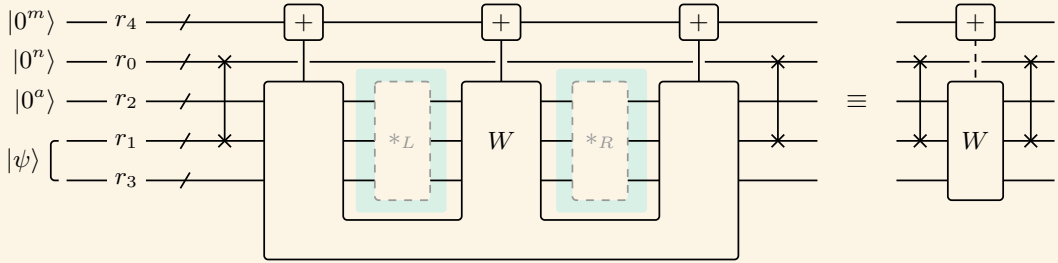
[1]This is a block diagonal block encoding containing $A \circ B$!

Recursive construction of element-wise products

Recursive construction of element-wise products

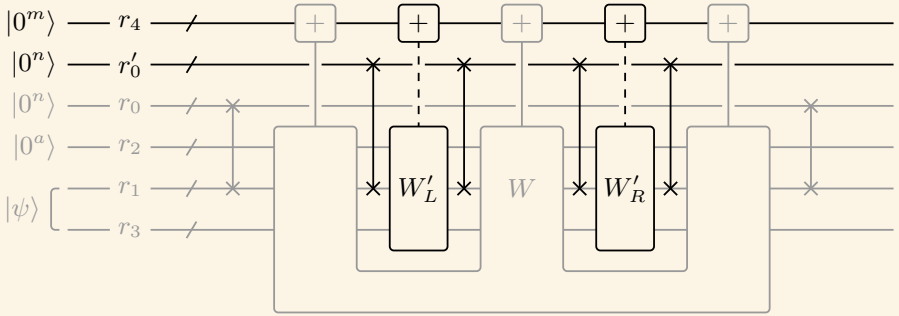


Recursive construction of element-wise products



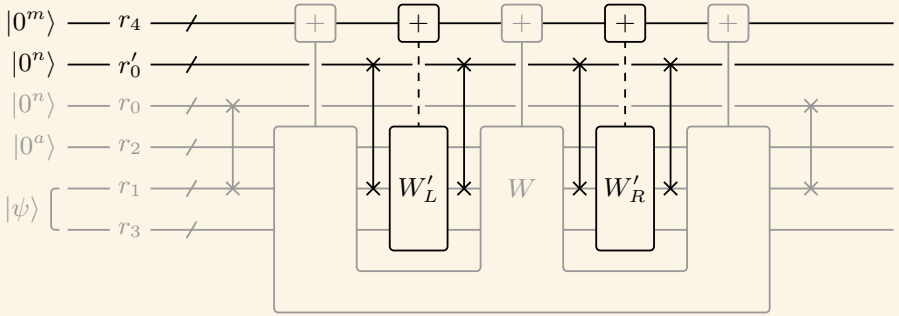
We can recast the base case as a two-slot function taking block encodings to block encodings with form $(I \otimes *)$

Recursive construction of element-wise products



Self-embedding circuit involves instantiating swap-copy register r'_0, and controlled additions to compression gadget register r_4

Recursive construction of element-wise products



Self-embedding circuit involves instantiating swap-copy register r'_0, and controlled additions to compression gadget register r_4

We recurse on both $*_R$, $*_L$ for $A^{\circ d}$ in $\log d$ depth (and thus aux space!)

Linear combination of unitaries

Linear combination of unitaries

We now have a quantum circuit for BEs of A^d for any d —

Linear combination of unitaries

We now have a quantum circuit for BEs of A^d for any d —

How do we build BEs of $f^\circ(A)$
for f a polynomial?

Linear combination of unitaries

We now have a quantum circuit for BEs of A^d for any d —

How do we build BEs of $f^\circ(A)$
for f a polynomial?

The usual approach is linear combination of unitaries (LCU) [CW12, BCK15], which relies on `select` unitary

Linear combination of unitaries

We now have a quantum circuit for BEs of A^d for any d —

How do we build BEs of $f^\circ(A)$
for f a polynomial?

The usual approach is linear combination of unitaries (LCU) [CW12, BCK15], which relies on `select` unitary

$$V = \sum_{k=1} |k\rangle\langle k| \otimes U_{A^{\circ k}} + \dots$$

Linear combination of unitaries

We now have a quantum circuit for BEs of A^d for any d —

How do we build BEs of $f^\circ(A)$
for f a polynomial?

The usual approach is linear combination of unitaries (LCU) [CW12, BCK15], which relies on `select` unitary

$$V = \sum_{k=1} |k\rangle\langle k| \otimes U_{A^{\circ k}} + \dots$$

This needs controlled BEs, and linear query complexity in maximum degree

LCU for the element-wise product — building `select`

LCU for the element-wise product — building `select`

If we build `select` for matrix products, then bit-wise controlled powers of two can suffice [CKS17]

$$U^{b_0} U^{2b_1} U^{2^2 b_2} \cdots = U^{\Sigma_k 2^k b_k}$$

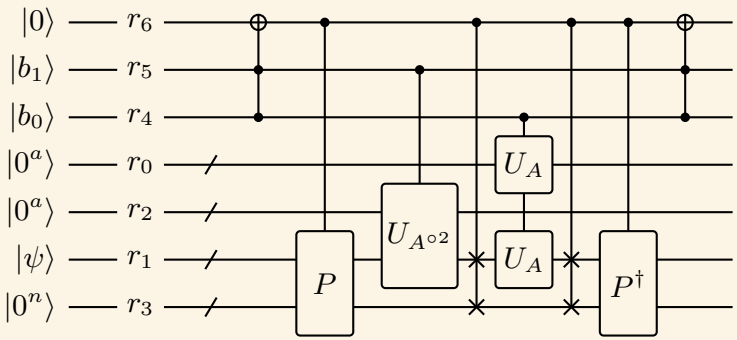
LCU for the element-wise product — building `select`

If we build `select` for matrix products, then bit-wise controlled powers of two can suffice [CKS17]

$$V = \sum_{k=1}^{d} |k\rangle\langle k|_{r_5} \otimes \left[U_{A^{\circ k}}\right]_{r_0 r_1 r_2 r_3 r_4} + \left[I_{r_5} - \sum_{k=1}^{d} |k\rangle\langle k|_{r_5}\right] \otimes I_{r_0 r_1 r_2 r_3 r_4}.$$

LCU for the element-wise product — building `select`

If we build `select` for matrix products, then bit-wise controlled powers of two can suffice [CKS17]



Note — the permutations and swaps are now controlled on the `XNOR` of the control bits; this reduces back to matrix product

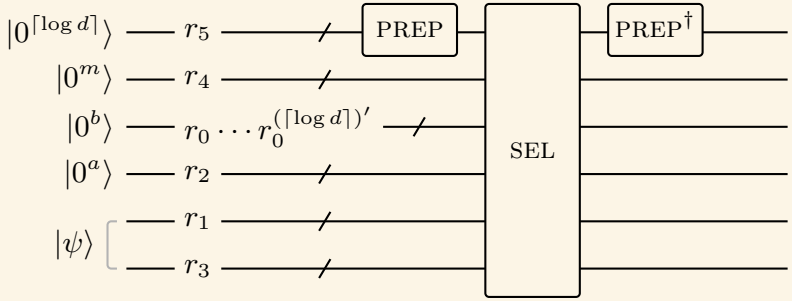
Putting it all together

Putting it all together

Given element-wise `select` unitary, standard LCU techniques [CW12] can achieve linear combinations of $A^{\circ k}$

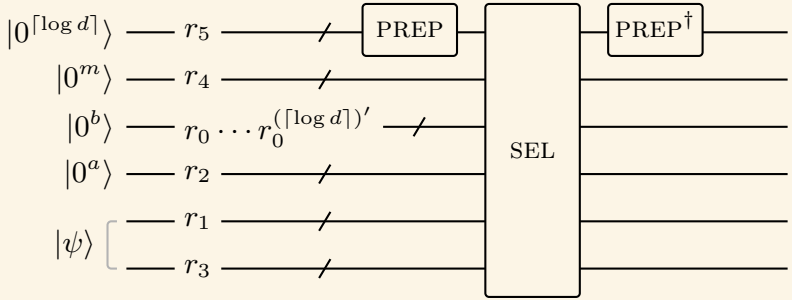
Putting it all together

Given element-wise `select` unitary, standard LCU techniques [CW12] can achieve linear combinations of $A^{\circ k}$



Putting it all together

Given element-wise `select` unitary, standard LCU techniques [CW12] can achieve linear combinations of $A^{\circ k}$



Degree d LCU requires $\log d$ space, and introduces subnormalization, error relating to ℓ_1 norm of coefficients …

Error propagation for block encoding methods

Error propagation for block encoding methods

Real BEs have subnormalization, aux space, and error; *how do these propagate through our algorithm?*

Error propagation for block encoding methods

Real BEs have subnormalization, aux space, and error; *how do these propagate through our algorithm?*

$\otimes$	`\otimes`	(Kronecker/tensor product),
$\circ$	`\circ`	(Element-wise product),
$\boxtimes$	`\boxtimes`	(Tracy–Singh product),
$\odot$	`\odot`	(Khatri-Rao product),
$*$	`\ast`	(Convolution).

Error propagation for block encoding methods

Real BEs have subnormalization, aux space, and error; *how do these propagate through our algorithm?*

$\otimes$	`\otimes`	(Kronecker/tensor product),
$\circ$	`\circ`	(Element-wise product),
$\boxtimes$	`\boxtimes`	(Tracy–Singh product),
$\odot$	`\odot`	(Khatri-Rao product),
$*$	`\ast`	(Convolution).

While element-wise product has unique properties, it is a submatrix of a matrix product, meaning subnorm and error propagation are bounded above by simple arguments

Amplified QEWTs

Amplified QEWTs

Let U_{A_k} be $(\alpha_k, a_k, \varepsilon_k \|A_k\|)$ block encodings of (possibly non-unitary) A_k for $k \in \{0, 1, \ldots, K-1\}$ and $\sum_k \varepsilon_k \leq 1/2$. Then for any $0 < \varepsilon < 1/2K$ we can construct an $(\alpha', a', \varepsilon')$ block encoding of the matrix product $A_{K-1} \cdots A_1 A_0$, where α' satisfies

Amplified QEWTs

Let U_{A_k} be $(\alpha_k, a_k, \varepsilon_k \|A_k\|)$ block encodings of (possibly non-unitary) A_k for $k \in \{0, 1, \ldots, K-1\}$ and $\sum_k \varepsilon_k \leq 1/2$. Then for any $0 < \varepsilon < 1/2K$ we can construct an $(\alpha', a', \varepsilon')$ block encoding of the matrix product $A_{K-1} \cdots A_1 A_0$, where α' satisfies

$$\frac{1}{2}(1-\delta)^{-L} \prod_k \|A_k\| \leq \alpha' \leq e^{1/2}(1-\delta)^{-L} \prod_k \|A_k\|,$$

Amplified QEWTs

Let U_{A_k} be $(\alpha_k, a_k, \varepsilon_k \|A_k\|)$ block encodings of (possibly non-unitary) A_k for $k \in \{0, 1, \ldots, K-1\}$ and $\sum_k \varepsilon_k \leq 1/2$. Then for any $0 < \varepsilon < 1/2K$ we can construct an $(\alpha', a', \varepsilon')$ block encoding of the matrix product $A_{K-1} \cdots A_1 A_0$, where α' satisfies

$$\frac{1}{2}(1-\delta)^{-L} \prod_k \|A_k\| \leq \alpha' \leq e^{1/2}(1-\delta)^{-L} \prod_k \|A_k\|,$$

and a' and ε' satisfy

$$a' = \max a_k + \log(K) + 2, \qquad \varepsilon' = e^{1/2}\left[K\varepsilon + \sum_k \varepsilon_k\right] \prod_k \|A_k\|.$$

Amplified QEWTs

Let U_{A_k} be $(\alpha_k, a_k, \varepsilon_k \|A_k\|)$ block encodings of (possibly non-unitary) A_k for $k \in \{0, 1, \ldots, K-1\}$ and $\sum_k \varepsilon_k \leq 1/2$. Then for any $0 < \varepsilon < 1/2K$ we can construct an $(\alpha', a', \varepsilon')$ block encoding of the matrix product $A_{K-1} \cdots A_1 A_0$, where α' satisfies

$$\frac{1}{2}(1-\delta)^{-L} \prod_k \|A_k\| \leq \alpha' \leq e^{1/2}(1-\delta)^{-L} \prod_k \|A_k\|,$$

and a' and ε' satisfy

$$a' = \max a_k + \log(K) + 2, \qquad \varepsilon' = e^{1/2}\left[K\varepsilon + \sum_k \varepsilon_k\right] \prod_k \|A_k\|.$$

Moreover, this amplification procedure uses $\mathcal{O}(\alpha_k/(\delta\|A_k\|) \log(\alpha_k/(\varepsilon\|A_k\|)))$ copies of each controlled-U_k and its inverse.

Applications

Common matrix functions

Common matrix functions

As an immediate consequence, we significantly improve space-use (and correctness) of machine-learning inference results [GYC+25]

Common matrix functions

As an immediate consequence, we significantly improve space-use (and correctness) of machine-learning inference results [GYC+25]

For matrix/vector softmax, appearing as core subroutine in self-attention for transformer model inference,

$$\text{softmax}(QK/\alpha)V, \quad \text{softmax}(x)_k \equiv e^{x_k} / \sum_j e^{x^j}$$

Common matrix functions

As an immediate consequence, we significantly improve space-use (and correctness) of machine-learning inference results [GYC+25]

For matrix/vector softmax, appearing as core subroutine in self-attention for transformer model inference,

$$\text{softmax}(QK/\alpha)V, \quad \text{softmax}(x)_k \equiv e^{x_k} / \sum_j e^{x^j}$$

we reduce auxiliary space from $n^2 \log[\varepsilon^{-1}]$ to $n \log(n \log[\varepsilon^{-1}])$

Common matrix functions

As an immediate consequence, we significantly improve space-use (and correctness) of machine-learning inference results [GYC+25]

For matrix/vector softmax, appearing as core subroutine in self-attention for transformer model inference,

$$\mathrm{softmax}(QK/\alpha)V, \quad \mathrm{softmax}(x)_k \equiv e^{x_k} / \sum_j e^{x^j}$$

we reduce auxiliary space from $n^2 \log[\varepsilon^{-1}]$ to $n \log(n \log[\varepsilon^{-1}])$

For common precisions, this is easily 15–20× space saving!

2D convolutions

2D convolutions

Element-wise products in Fourier basis correspond to convolution in standard basis — this holds for tensors! [HJ85]

2D convolutions

Element-wise products in Fourier basis correspond to convolution in standard basis — this holds for tensors! [HJ85]

$$(h * g)_{\mathrm{circ}}(k_1, k_2) \equiv \sum_{j_1=0}^{K-1} \sum_{j_2=0}^{K-1} h(j_1, j_2) g(k_1 - j_1 \pmod K, k_2 - j_2 \pmod K),$$

2D convolutions

Element-wise products in Fourier basis correspond to convolution in standard basis — this holds for tensors! [HJ85]

$$(h * g)_{\text{circ}}(k_1, k_2) \equiv \sum_{j_1=0}^{K-1} \sum_{j_2=0}^{K-1} h(j_1, j_2) g(k_1 - j_1 \pmod K, k_2 - j_2 \pmod K),$$

$$(h * g)_{\text{circ}}(k_1, k_2) = \mathscr{F}^{-1}(H \circ G) = \mathscr{F}^{-1}(\mathscr{F}(h) \circ \mathscr{F}(g)),$$

2D convolutions

Element-wise products in Fourier basis correspond to convolution in standard basis — this holds for tensors! [HJ85]

$$(h * g)_{\text{circ}}(k_1, k_2) \equiv \sum_{j_1=0}^{K-1} \sum_{j_2=0}^{K-1} h(j_1, j_2) g(k_1 - j_1 \pmod K, k_2 - j_2 \pmod K),$$

$$(h * g)_{\text{circ}}(k_1, k_2) = \mathscr{F}^{-1}(H \circ G) = \mathscr{F}^{-1}(\mathscr{F}(h) \circ \mathscr{F}(g)),$$

$$(h * g)_{\text{circ}} = F^{\dagger}([FhF^T] \circ [FgF^T])F^* = F^{\dagger}([FhF] \circ [FgF])F^{\dagger},$$

2D convolutions

Element-wise products in Fourier basis correspond to convolution in standard basis — this holds for tensors! [HJ85]

$$(h * g)_{\mathrm{circ}}(k_1, k_2) \equiv \sum_{j_1=0}^{K-1} \sum_{j_2=0}^{K-1} h(j_1, j_2) g(k_1 - j_1 \pmod K, k_2 - j_2 \pmod K),$$

$$(h * g)_{\mathrm{circ}}(k_1, k_2) = \mathscr{F}^{-1}(H \circ G) = \mathscr{F}^{-1}(\mathscr{F}(h) \circ \mathscr{F}(g)),$$

$$(h * g)_{\mathrm{circ}} = F^{\dagger}([FhF^T] \circ [FgF^T])F^* = F^{\dagger}([FhF] \circ [FgF])F^{\dagger},$$

This gives a depth $n^2 \log n$ algorithm for computing circular convolution: (α, a, ε), $(\beta, b, \delta) \mapsto (\alpha\beta, \max(a, b) + 1, \alpha\delta + \beta\varepsilon + \varepsilon\delta)$

Non-standard matrix products

Non-standard matrix products

∃ block analogues of Kronecker & element-wise products (Khatri-Rao [KR68] and Tracy-Singh [TS72]) that appear in signal processing/tensor decomp

Non-standard matrix products

$\exists$ block analogues of Kronecker & element-wise products (Khatri-Rao [KR68] and Tracy-Singh [TS72]) that appear in signal processing/tensor decomp

$$(M_1 \boxtimes M_2) \equiv \left[\begin{array}{cc|cc} A_{00} \otimes B_{00} & A_{00} \otimes B_{01} & A_{01} \otimes B_{00} & A_{01} \otimes B_{01} \\ A_{00} \otimes B_{10} & A_{00} \otimes B_{11} & A_{01} \otimes B_{10} & A_{01} \otimes B_{11} \\ \hline A_{10} \otimes B_{00} & A_{10} \otimes B_{01} & A_{11} \otimes B_{00} & A_{11} \otimes B_{01} \\ A_{10} \otimes B_{10} & A_{10} \otimes B_{11} & A_{11} \otimes B_{10} & A_{11} \otimes B_{11} \end{array}\right],$$

Non-standard matrix products

∃ block analogues of Kronecker & element-wise products (Khatri-Rao [KR68] and Tracy-Singh [TS72]) that appear in signal processing/tensor decomp

$$(M_1 \boxtimes M_2) \equiv \left[\begin{array}{cc|cc} A_{00} \otimes B_{00} & A_{00} \otimes B_{01} & A_{01} \otimes B_{00} & A_{01} \otimes B_{01} \\ A_{00} \otimes B_{10} & A_{00} \otimes B_{11} & A_{01} \otimes B_{10} & A_{01} \otimes B_{11} \\ \hline A_{10} \otimes B_{00} & A_{10} \otimes B_{01} & A_{11} \otimes B_{00} & A_{11} \otimes B_{01} \\ A_{10} \otimes B_{10} & A_{10} \otimes B_{11} & A_{11} \otimes B_{10} & A_{11} \otimes B_{11} \end{array}\right],$$

$$(M_1 \odot M_2) = \left[\begin{array}{c|c} A_{00} \otimes B_{00} & A_{01} \otimes B_{01} \\ \hline A_{10} \otimes B_{10} & A_{11} \otimes B_{11} \end{array}\right].$$

Non-standard matrix products

∃ block analogues of Kronecker & element-wise products (Khatri-Rao [KR68] and Tracy-Singh [TS72]) that appear in signal processing/tensor decomp

$$(M_1 \boxtimes M_2) \equiv \left[\begin{array}{cc|cc} A_{00}\otimes B_{00} & A_{00}\otimes B_{01} & A_{01}\otimes B_{00} & A_{01}\otimes B_{01} \\ A_{00}\otimes B_{10} & A_{00}\otimes B_{11} & A_{01}\otimes B_{10} & A_{01}\otimes B_{11} \\ \hline A_{10}\otimes B_{00} & A_{10}\otimes B_{01} & A_{11}\otimes B_{00} & A_{11}\otimes B_{01} \\ A_{10}\otimes B_{10} & A_{10}\otimes B_{11} & A_{11}\otimes B_{10} & A_{11}\otimes B_{11} \end{array}\right],$$

$$(M_1 \odot M_2) = \left[\begin{array}{c|c} A_{00}\otimes B_{00} & A_{01}\otimes B_{01} \\ \hline A_{10}\otimes B_{10} & A_{11}\otimes B_{11} \end{array}\right].$$

We provide simple quantum algorithms for these products/their hybrids with LCU and QSVT, all of them space-efficient

Outlook and extensions

Outlook and extensions

We have a query-optimal protocol for QEWTs that uses only logarithmically, in d, more space than the diagonal (QSVT) setting

Outlook and extensions

We have a query-optimal protocol for QEWTs that uses only logarithmically, in d, more space than the diagonal (QSVT) setting

Reducing space use in low rank, near-diagonal, circulant, Hadamard, or approximate settings …

Outlook and extensions

We have a query-optimal protocol for QEWTs that uses only logarithmically, in d, more space than the diagonal (QSVT) setting

Reducing space use in low rank, near-diagonal, circulant, Hadamard, or approximate settings …

Further end-to-end applications for signal processing [HJ91], machine learning primitives [CWP+25], and statistical inference [RHG+25]

Outlook and extensions

We have a query-optimal protocol for QEWTs that uses only logarithmically, in d, more space than the diagonal (QSVT) setting

Reducing space use in low rank, near-diagonal, circulant, Hadamard, or approximate settings …

Further end-to-end applications for signal processing [HJ91], machine learning primitives [CWP+25], and statistical inference [RHG+25]

Incorporating randomization [MR25] parallelization [MRC+25] to further optimize to specific hardware

🏠 Take-home

🏠 Take-home

We establish a firm foundation for the theory of quantum element-wise transforms (QEWTs)

🏠 Take-home

We establish a firm foundation for the theory of quantum element-wise transforms (QEWTs), rectifying errors in prior work

🏠 Take-home

We establish a firm foundation for the theory of quantum element-wise transforms (QEWTs), rectifying errors in prior work, and reducing space use exponentially in the degree of the applied function[1]

[1] Comparable in cost to other BE algorithms!

🏠 Take-home

We establish a firm foundation for the theory of quantum element-wise transforms (QEWTs), rectifying errors in prior work, and reducing space use exponentially in the degree of the applied function[1]

Fundamentally distinct from QSVT or LCU, relying on novel subroutines for space-catalytically transforming block encodings with block structure

[1] Comparable in cost to other BE algorithms!

🏠 Take-home

We establish a firm foundation for the theory of quantum element-wise transforms (QEWTs), rectifying errors in prior work, and reducing space use exponentially in the degree of the applied function[1]

Fundamentally distinct from QSVT or LCU, relying on novel subroutines for space-catalytically transforming block encodings with block structure

Cheap nonlinear transform for high-dim data, ubiquitous to machine learning, signal processing, and statistics

[1] Comparable in cost to other BE algorithms!

🏠 Take-home

We establish a firm foundation for the theory of quantum element-wise transforms (QEWTs), rectifying errors in prior work, and reducing space use exponentially in the degree of the applied function[1]

Quantum algorithms for numerical linear algebra are only cursorily understood, and rapidly developing!

[1] Comparable in cost to other BE algorithms!

[ACL23] Dong An, Andrew M. Childs, and Lin Lin. Quantum algorithm for linear non-unitary dynamics with near-optimal dependence on all parameters, 2023.

[ALM+24] Michel Alexis, Lin Lin, Gevorg Mnatsakanyan, Christoph Thiele, and Jiasu Wang. Infinite quantum signal processing for arbitrary szegö functions, 2024.

[BCK15] Dominic W Berry, Andrew M Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th annual symposium on foundations of computer science*, pages 792–809. IEEE, 2015.

[CAS+22] Pedro C.S. Costa, Dong An, Yuval R. Sanders, Yuan Su, Ryan Babbush, and Dominic W. Berry. Optimal scaling quantum linear-systems solver via discrete adiabatic theorem. *PRX Quantum*, 3:040303, 2022.

[CDG+20] Rui Chao, Dawei Ding, Andras Gilyen, Cupjin Huang, and Mario Szegedy. Finding angles for quantum signal processing with machine precision. *arXiv preprint arXiv:2003.02831*, 2020.

[CGL+20] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 387–400, New York, NY, USA, 2020. Association for Computing Machinery.

[CKS17] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, Jan 2017.

[CW12] Andrew M. Childs and Nathan Wiebe. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Info. Comput.*, 12(11–12):901–924, Nov 2012.

[CWP+25] Grigorios G. Chrysos, Yongtao Wu, Razvan Pascanu, Philip H.S. Torr, and Volkan Cevher. Hadamard product in deep learning: Introduction, advances and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(8):6531–6549, 2025.

[DLNW24a] Yulong Dong, Lin Lin, Hongkang Ni, and Jiasu Wang. Infinite quantum signal processing. *Quantum*, 8:1558, 2024.

[DLNW24b] Yulong Dong, Lin Lin, Hongkang Ni, and Jiasu Wang. Robust iterative method for symmetric quantum signal processing in all parameter regimes. *SIAM Journal on Scientific Computing*, 46(5):A2951–A2971, 2024.

[DMWL21] Yulong Dong, Xiang Meng, K. Birgitta Whaley, and Lin Lin. Efficient phase-factor evaluation in quantum signal processing. *Phys. Rev. A*, 103(4), 2021.

[GLG23] Sevag Gharibian and François Le Gall. Dequantizing the quantum singular value transformation: Hardness and applications to quantum chemistry and the quantum pcp conjecture. *SIAM Journal on Computing*, 52(4):1009–1038, August 2023.

[GLM+22] András Gilyén, Seth Lloyd, Iman Marvian, Yihui Quek, and Mark M. Wilde. Quantum algorithm for petz recovery channels and pretty good measurements. *Physical Review Letters*, 128(22), 2022.

[GMF24] Naixu Guo, Kosuke Mitarai, and Keisuke Fujii. Nonlinear transformation of complex amplitudes via quantum singular value transformation. *Phys. Rev. Res.*, 6:043227, Dec 2024.

[GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, 2019.

[GYC+25] Naixu Guo, Zhan Yu, Matthew Choi, Yizhan Han, Aman Agrawal, Kouhei Nakaji, Alán Aspuru-Guzik, and Patrick Rebentrost. Quantum transformer: Accelerating model inference via quantum linear algebra, 2025.

[Haa19] Jeongwan Haah. Product decomposition of periodic functions in quantum signal processing. *Quantum*, 3:190, Oct 2019.

[HCSS23] Zoë Holmes, Nolan J. Coble, Andrew T. Sornborger, and Subaşı. Nonlinear transformations in quantum computation. *Phys. Rev. Res.*, 5:013105, Feb 2023.

[HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009.

[HJ85] Roger A. Horn and Charles R. Johnson. ***Matrix Analysis***. Cambridge University Press, 1985.

[HJ91] Roger A. Horn and Charles R. Johnson. ***Topics in Matrix Analysis***. Cambridge University Press, 1991.

[KLB+25] Robbie King, Guang Hao Low, Ryan Babbush, Rolando D. Somma, and Nicholas C. Rubin. Quantum simulation with sum-of-squares spectral amplification, 2025.

[KR68] C. G. Khatri and C. Radhakrishna Rao. Solutions to some functional equations and their applications to characterization of probability distributions. *Sankhyā Ser. A*, 30(2):167–180, 1968.

[LC17] G. H. Low and I. L. Chuang. Optimal Hamiltonian simulation by quantum signal processing. *Phys. Rev. Lett.*, 118:010501, 2017.

[LC19] G. H. Low and I. L. Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, 2019.

[Lin25] Lin Lin. Mathematical and numerical analysis of quantum signal processing, 2025.

[LKAS+21] Seth Lloyd, Bobak T. Kiani, David R. M. Arvidsson-Shukur, Samuel Bosch, Giacomo De Palma, William M. Kaminsky, Zi-Wen Liu, and Milad Marvian. Hamiltonian singular value transformation and inverse block encoding. *arXiv preprint arXiv:2104.01410*, 2021.

[LS24] Guang Hao Low and Yuan Su. Quantum eigenvalue processing. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, page 1051–1062. IEEE, October 2024.

[LS25] Guang Hao Low and Rolando D. Somma. Optimal quantum simulation of linear non-unitary dynamics, 2025.

[LT20] Lin Lin and Yu Tong. Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems. *Quantum*, 4:361, November 2020.

[LW19] Guang Hao Low and Nathan Wiebe. Hamiltonian simulation in the interaction picture. *arXiv preprint, arXiv:1805.00675*, 2019.

[LYC14] Guang Hao Low, Theodore J Yoder, and Isaac L Chuang. Optimal arbitrarily accurate composite pulse sequences. *Phys. Rev. A*, 89(2):022341, 2014.

[LYC16] G. H. Low, T. J. Yoder, and I. L. Chuang. Methodology of resonant equiangular composite quantum gates. *Phys. Rev. X*, 6:041067, 2016.

[MA24] C. Thiele M. Alexis, G. Mnatsakanyan. Quantum signal processing and nonlinear Fourier analysis. *Rev. Mat. Complut.*, pages 655–694, 2024.

[MLCC23] John M. Martyn, Yuan Liu, Zachary E. Chin, and Isaac L. Chuang. Efficient fully-coherent quantum signal processing algorithms for real-time dynamics simulation. *The Journal of Chemical Physics*, 158(2), 2023.

[MR25] John M. Martyn and Patrick Rall. Halving the cost of quantum algorithms with randomization. *npj Quantum Inf.*, 11(47), 2025.

[MRC+25] John M. Martyn, Zane M. Rossi, Kevin Z. Cheng, Yuan Liu, and Isaac L. Chuang. Parallel quantum signal processing via polynomial factorization. *Quantum*, 9:1834, 2025.

[MRTC21] John M Martyn, Zane M Rossi, Andrew K Tan, and Isaac L Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2(4):040203, 2021.

[MW24] Danial Motlagh and Nathan Wiebe. Generalized quantum signal processing. *PRX Quantum*, 5:020368, 2024.

[NSYL25] Hongkang Ni, Rahul Sarkar, Lexing Ying, and Lin Lin. Inverse nonlinear fast fourier transform on su(2) with applications to quantum signal processing, 2025.

[Ral21] Patrick Rall. Faster Coherent Quantum Algorithms for Phase, Energy, and Amplitude Estimation. *Quantum*, 5:566, October 2021.

[RBMC23] Zane M. Rossi, Victor M. Bastidas, William J. Munro, and Isaac L. Chuang. Quantum signal processing with continuous variables, 2023.

[RC22] Zane M. Rossi and Isaac L. Chuang. Multivariable quantum signal processing (M-QSP): prophecies of the two-headed oracle. *Quantum*, 6:811, Sep 2022.

[RCC25] Zane M. Rossi, Jack L. Ceroni, and Isaac L. Chuang. Modular quantum signal processing in many variables. *Quantum*, 9:1776, June 2025.

[RHG+25] Arthur G. Rattew, Po-Wei Huang, Naixu Guo, Lirandë Pira, and Patrick Rebentrost. Accelerating inference for multilayer neural networks with quantum computers. *arXiv preprint, arXiv:2510.07195*, 2025.

[Ros25] Zane M. Rossi. A Solovay–Kitaev theorem for quantum signal processing, 2025.

[RR23] Arthur G. Rattew and Patrick Rebentrost. Non-linear transformations of quantum amplitudes: Exponential improvement, generalization, and applications. *arXiv preprint, arXiv:2309.09839*, 2023.

[SLBB25] Rolando D. Somma, Guang Hao Low, Dominic W. Berry, and Ryan Babbush. Quantum algorithm for linear matrix equations, 2025.

[Sty73] George P.H. Styan. Hadamard products and multivariate statistical analysis. *Linear Algebra and its Applications*, 6:217–240, 1973.

[TB22] Lloyd N. Trefethen and David Bau. ***Numerical Linear Algebra, Twenty-fifth Anniversary Edition***. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2022.

[Tre19] Lloyd N Trefethen. ***Approximation Theory and Approximation Practice, Extended Edition***. SIAM, 2019.

[TS72] Derrick S. Tracy and Rana P. Singh. A new matrix product and its applications in partitioned matrix differentiation. *Statistica Neerlandica*, 26(4):143–157, 1972.

[TT23] Ewin Tang and Kevin Tian. A CS guide to the quantum singular value transformation. *arXiv preprint, arXiv:2302.14324*, 2023.

[VC05] Lieven MK Vandersypen and Isaac L Chuang. NMR techniques for quantum control and computation. *RMP*, 76(4):1037, 2005.

[WDL22] Jiasu Wang, Yulong Dong, and Lin Lin. On the energy landscape of symmetric quantum signal processing. *Quantum*, 6:850, 2022.

[Wim94] Stephen Wimperis. Broadband, narrowband, and passband composite pulses for use in advanced NMR experiments. *Journal of Magnetic Resonance, Series A*, 109(2):221–231, 1994.

[YLC14] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang. Fixed-point quantum search with an optimal number of queries. *Physical Review Letters*, 113(21), 2014.

Bonus slides