10

Quantum linear algebra

At a high level of abstraction, quantum computers compose unitary matrices, and do so with classically unparalleled efficiency. This hints at quantum speedups for linear algebra tasks. However, often one needs to work with large non-unitary matrices; thus, for performing general linear algebra tasks we often wish to embed certain non-unitary matrices into unitary matrices represented by efficient quantum circuits, and then apply them to quantum states, take their sums or products, or implement more general matrix functions. These tasks are collectively referred to as "quantum linear algebra," the building blocks of which are discussed in this chapter.

The techniques described in this chapter evolved over the past decades and converged to the presented unified framework within several distinct research threads. Block-encodings emerged as a natural approach for embedding non-unitary matrices into quantum circuits, inspired by approaches based on purification, dilation (e.g., Stinespring representation [1050] or Stinespring dilation [1044]), and postselection. Quantum signal processing (QSP) was discovered as a byproduct of the characterization of simple single-qubit pulse sequences used in nuclear magnetic resonance [719], for synthesizing polynomial transformations applicable to a "signal parameter" encoded as a matrix element of a single-qubit rotation matrix. Meanwhile, it was extensively studied how matrix functions could be synthesized using the linear combinations of unitaries technique on matrix exponentials implemented by Hamiltonian simulation [281, 48, 248], or Chebyshev polynomials of operators implemented via quantum walk techniques [135, 136, 282]. Such matrix exponentials or Chebyshev polynomials can be implemented, for example, via qubitization of a block-encoded operator. In parallel to progress on advanced amplitude amplification [466, 1067] techniques, it was recognized [716, 717] that QSP can be "lifted" for applying polynomial transformations to the eigenvalues of quantum walk operators (such as those implemented by qubitization), and thus for

implementing a rich family of matrix functions, immediately yielding an optimal algorithm for time-independent Hamiltonian simulation. The concepts of qubitization and QSP were later generalized and unified into the framework of quantum singular value transformation [431], providing generalizations and more efficient implementations of a number of existing quantum algorithms and leading to the discovery of several new algorithms.

The authors are grateful to Lin Lin for reviewing this chapter.

10.1 Block-encodings

Rough overview (in words)

In a quantum algorithm, the quantum gates that are applied to quantum states are necessarily unitary operators. However, one often needs to apply a linear transformation to some encoded data that is not represented by a unitary operator, and furthermore, one generally needs coherent access to these non-unitary transformations. How can we encode such a non-unitary transformation within a unitary operator? Block-encoding is one method of providing exactly this kind of coherent access to generic linear operators. Block-encoding works by embedding the desired linear operator as a suitably normalized block within a larger unitary matrix, such that the full encoding is a unitary operator, and the desired linear operator is given by restricting the unitary to an easily recognizable subspace. To be useful for quantum algorithms, this block-encoding unitary must also be realized by some specific quantum circuit acting on the main register and additional ancilla qubits.

Block-encodings are ubiquitous within quantum algorithms, but they have both benefits and drawbacks. They are easy to work with, since one can efficiently perform manipulations of block-encodings, such as taking products or convex combinations. On the other hand, this improved working efficiency comes at the cost of having more limited access. For example, if a matrix is stored in classical random access memory, the matrix entries can be explicitly accessed with a single query to the memory, whereas if one only has access to a block-encoding of the matrix, estimating a matrix entry to precision ε requires $O(1/\varepsilon)$ uses of the block-encoding unitary in general (by utilizing an amplitude estimation subroutine).

Block-encodings also provide a layer of abstraction that assists in the design and analysis of quantum algorithms. One can simply assume access to a block-encoding and count the number times it is applied. To run the algorithm, it is

necessary to choose a method for implementing the block-encoding. There are many ways of constructing block-encodings that could be suited to the structure of the input. For instance, there are efficient block-encoding strategies for density matrices, positive operator-valued measures (POVMs), Gram matrices, sparse matrices, matrices that are stored in quantum data structures, structured matrices, and operators given as a linear combination of unitaries (with a known implementation). We discuss these constructions below. For unstructured, dense matrices, the strategy for Gram matrices can be instantiated using state preparation and quantum random access memory (QRAM) as subroutines. For more details on a particular block-encoding scheme for loading matrices of classical data, see Section 17.3 on block-encoding dense matrices of classical data.

Rough overview (in math)

Our goal is to build a unitary operator that gives coherent access to an $M \times M$ matrix A (we will later relax the assumption that A is square), with normalization $\alpha \geq \|A\|$, where $\|A\|$ denotes the spectral norm of A. As the name suggests, block-encoding is a way of encoding the matrix A as a block in a larger unitary matrix

$$U_A = \begin{array}{c} \\ |0^a\rangle \\ |0^a\rangle_\perp \end{array} \begin{array}{c} \begin{array}{cc} |0^a\rangle & |0^a\rangle_\perp \end{array} \\ \begin{pmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{pmatrix} \end{array},$$

where the labels $|0^a\rangle$ and $|0^a\rangle_\perp$ indicate which portion of the vector space each block corresponds to—specifically, whether the first a qubits are equal to or orthogonal to the state $|0^a\rangle$, respectively. Three of the four blocks are unspecified and can take on any values such that U_A is unitary. More precisely, we say that the unitary U_A is an (α, a, ϵ)-block-encoding of the matrix $A \in \mathbb{C}^{M \times M}$ if

$$\|A - \alpha(\langle 0^a| \otimes I) U_A (|0^a\rangle \otimes I)\| \leq \epsilon, \tag{10.1}$$

where $a \in \mathbb{N}$ is the number of ancilla qubits used for embedding the block-encoded operator, and $\alpha, \epsilon \in \mathbb{R}_+$ define the normalization and error, respectively. Note that $\alpha \geq \|A\| - \epsilon$ is necessary for U_A to be unitary. The definition above can be extended for general matrices, though additional embedding or padding may be needed (e.g., to make the matrix square).

Once a block-encoding is constructed, it can be used in a quantum algorithm to apply the matrix A to a quantum state by applying the unitary U_A to the larger quantum system. The application of the block-encoding can be thought of as a probabilistic application of A—applying U_A to $|0^a\rangle|\psi\rangle$ and postselecting on the

first register being in the state $|0^a\rangle$ gives an output state proportional to $A|\psi\rangle$ in the second register.

There are several ways of implementing block-encodings based on the choice of matrix A [431, Section 4.2].[1]

- Unitary matrices are $(1, 0, 0)$-block-encodings of themselves. Controlled unitaries (e.g., CNOT) are essentially $(1, 1, 0)$-block-encodings of the controlled operation.
- Given an s-qubit density matrix ρ and an $(a+s)$-qubit unitary G that prepares a *purification* of ρ as $G|0^a\rangle|0^s\rangle = |\rho\rangle$ (s.t. $\mathrm{tr}_a|\rho\rangle\langle\rho| = \rho$, where tr_a denotes trace over the first register), then the operator [717]

$$(G^\dagger \otimes I_s)(I_a \otimes \mathrm{SWAP}_s)(G \otimes I_s)$$

  is a $(1, a + s, 0)$-block-encoding of the density matrix ρ, where I_x denotes the identity operator on a register with x qubits, and SWAP_s denotes the operation that swaps two s-qubit registers [431, Lemma 45].
- Similarly, one can construct block-encodings of POVM operators, given access to a unitary that implements the POVM [45]. Specifically, if U is a unitary that implements the POVM M to precision ϵ, such that for all s-qubit density operators ρ we have

$$\left|\mathrm{tr}(\rho M) - \mathrm{tr}\Big[U(|0\rangle\langle 0|^{\otimes a} \otimes \rho)U^\dagger(|0\rangle\langle 0| \otimes I_{a+s-1}))\Big]\right| \leq \epsilon,$$

  then $(I_1 \otimes U^\dagger)(\mathrm{CNOT} \otimes I_{a+s-1})(I_1 \otimes U)$ is a $(1, 1 + a, \epsilon)$-block-encoding of M [431, Lemma 46].
- One can also implement a block-encoding of a Gram matrix using a pair of state preparation unitaries U_L and U_R. In particular, the product

$$U_A = U_L^\dagger U_R$$

  is a $(1, a, 0)$-block-encoding of the Gram matrix A whose entries are $A_{ij} = \langle\psi_i|\phi_j\rangle$, where [431, Lemma 47]

$$U_L|0^a\rangle|i\rangle = |\psi_i\rangle, \qquad U_R|0^a\rangle|j\rangle = |\phi_j\rangle.$$

- One can generalize the above strategy from Gram matrices to arbitrary matrices to produce (α, a, ϵ)-block-encodings of general matrices A, where again $\alpha \geq \|A\|$. See Section 17.3 on block-encoding dense matrices of classical data for details.
- Sparse matrices: Given a matrix $A \in \mathbb{C}^{2^w \times 2^w}$ that is s_r-row sparse and s_c-column sparse (meaning each row and column has at most s_r and s_c nonzero

[1] References to locations in [431] typically refer to the longer arXiv version, rather than the STOC version.

entries, respectively), then, defining $\|A\|_{\max} = \max_{i,j} |A_{ij}|$, one can create a $(\sqrt{s_r s_c}\|A\|_{\max}, w+3, \epsilon)$-block-encoding of A using oracles O_r, O_c, and O_A, defined as follows [431, Lemma 48]

$$\begin{aligned} O_r &: |i\rangle|k\rangle \mapsto |i\rangle|r_{ik}\rangle, && \forall i \in [2^w]-1, k \in [s_r], \\ O_c &: |\ell\rangle|j\rangle \mapsto |c_{\ell j}\rangle|j\rangle, && \forall \ell \in [s_c], j \in [2^w]-1, \\ O_A &: |i\rangle|j\rangle|0^b\rangle \mapsto |i\rangle|j\rangle|A_{ij}\rangle, && \forall i, j \in [2^w]-1. \end{aligned}$$

In the above, r_{ij} is the index of the j-th nonzero entry in the i-th row of A (or $j + 2^w$ if there are less than i nonzero entries), c_{ij} is the index of the i-th nonzero entry in the j-th column of A (or $i + 2^w$ if there are less than j nonzero entries), and $|A_{ij}\rangle$ is a b-bit binary encoding of the matrix element A_{ij}. To build the block-encoding, one needs one query to each of O_r and O_c, and two queries of O_A. This input model is known as the sparse access model. If, in addition to being sparse, the matrix also enjoys some additional *structure*, for example, there are only a few distinct values that the matrix elements can take, the complexity can be further improved [969, 227]. Finally, note that the sparsity dependence can be essentially quadratically improved—reducing the block-encoding normalization factor from $\sqrt{s_r s_c}\|A\|_{\max}$ to $(\max(s_r, s_c))^{(1+o(1))/2}\|A\|_{1\to 2}$, where $\|A\|_{1\to 2} = \max_v \|Av\|_2/\|v\|_1$—using advanced Hamiltonian simulation techniques [714, Theorem 2] combined with taking the logarithm of unitaries [431, Corollary 71], however, the resulting subroutine may be impractical and comes with a worse precision dependence.

- For matrices given as a linear combination of unitary operators (LCU), we can block-encode the matrix using the LCU technique [281]. We provide a full description in §Linear combinations of Section 10.2, and only give a brief outline here. For $A = \sum_{i=1}^{L} c_i V_i$ with V_i unitary, we define the oracles PREPARE (acting on $\lceil \log_2(L) \rceil$ ancilla qubits) and SELECT (acting on the ancilla and register qubits), and implement a $(\sum_i |c_i|, \lceil \log_2(L) \rceil, 0)$-block-encoding of A, using $U := \text{PREPARE}^\dagger \cdot \text{SELECT} \cdot \text{PREPARE}$. The Hamiltonians of physical systems can often be written as a linear combination of a moderate number of Pauli operators, leading to a prevalence of this technique in quantum algorithms for chemistry [75, 140] and condensed matter physics [75, 283, 1011].

In addition to the definition of block-encoding in Eq. (10.1), one can also define an asymmetric version as follows

$$\left\| A - \alpha(\langle 0^a| \otimes I) U_A (|0^b\rangle \otimes I) \right\| \leq \epsilon,$$

where a may not equal b. In this case, U_A can be considered to be an $(\alpha, (a, b), \epsilon)$- or an $(\alpha, \max(a, b), \epsilon)$-block-encoding of A. This can be useful for block-encoding a non-square matrix.

Dominant resource cost (gates/qubits)

The complexity of block-encoding an operator depends on the type of data or operator being encoded and any underlying assumptions. For instance, unitaries are naturally block-encodings of themselves, and hence their resource requirements depend entirely on their circuit-level implementation without any additional overhead for being a "block-encoding." By contrast, approaches that make use of state preparation and QRAM to implement the block-encoding tend to have larger complexities, as those two subroutines typically dominate the resource requirements. For example, the best known circuits that implement block-encodings of matrices of classical data for general, dense $N \times N$ matrices use $O(N \log(1/\epsilon))$ qubits to achieve minimum T-gate count (which also scales as $O(N \log(1/\epsilon))$), or a larger $O(N^2)$ number of qubits to achieve minimum T-gate depth (which scales as $O(\log(N) + \log(1/\epsilon))$ [296]. In the sparse access model, one can use $O(w + \log^{2.5}(s_r s_c/\epsilon))$ one- and two-qubit gates, and $O(b + \log^{2.5}(s_r s_c/\epsilon))$ ancilla qubits [431], in addition to the calls to the matrix entry O_A and sparse access oracles O_r and O_c, which must be implemented either by computing matrix entries "on-the-fly" or by using a primitive (see [1084] for asymptotic resource statements for general sparse matrices with varying ancilla availability). Assuming appropriate binary representations of the numbers A_{ij}, the exponents of the above logarithms can be reduced to 1 using the techniques of [894] (see also [75, Section III.D] and [212, Supplementary Material VII.A.2]).

The value of block-encodings is not that it is always cheap to implement them (as it depends on the relevant cost metric and the data access model); rather, the concept of block-encodings is powerful because it allows a practitioner of quantum algorithms to study and optimize the block-encoding construction independently of how it is used within the larger algorithm.

Caveats

The definition of block-encoding requires $\|A\|/\alpha \leq 1$. If $\|A\|/\alpha = 1$, then the block-encoding achieves an optimal normalization factor α. However, note that often the above constructions lead to suboptimal normalization factors in the sense that $\alpha \gg \|A\|$. In practical applications, this suboptimality usually leads to a corresponding increase in the overall complexity.

For a given desired block-encoding, there can be several independent, yet equally valid implementations, and one can sometimes optimize for various re-

sources when building the block-encoding. For example, many block-encoding strategies require a step in which some classical data is loaded into QRAM, but there are several ways of performing this data-loading step.

When using a block-encoding as part of a larger quantum algorithm, it is important to ensure that the overhead introduced by implementing a block-encoding will not outweigh any potential quantum speedups, as block-encoding can be very resource intensive.

The use of $|0\rangle^{\otimes a}$ as the "signal" state is just one convention—we can use any "signal" state, given a unitary to prepare it [717]. One can also consider a more general definition known as "projected unitary encodings" which allows using an arbitrary subspace, rather than just a state-indexed block [431].

Example use cases

Block-encodings are ubiquitous in quantum algorithms, and they prevail in quantum algorithms that need coherent access to some linear operator or a method of implementing a non-unitary transformation on quantum data. Some specific examples:

- We can manipulate block-encoded operators—for example, take convex or linear combinations, products, tensor products, and other transformations of an input operator.
- The combination of qubitization with quantum signal processing, or quantum singular value transformation can be used to realize algorithms by applying polynomial transformations to block-encoded matrices. Prominent examples are Hamiltonian simulation via qubitization, and matrix (pseudo) inversion [431, Theorem 41] that can be used for solving large linear systems of equations [500] or more generally least-squares regression problems [248].
- Block-encoding can be used to provide coherent access to classical data in a quantum algorithm; for example, loading classical data into a quantum interior point method for portfolio optimization [328].

Further reading

Reference [248] provides an instructive overview of the concept of block-encoding and showcases its power in several applications related to (generalized) regression problems. Meanwhile, [431] is a comprehensive collection of technical results about block-encodings and quantum linear algebra more generally.

10.2 Manipulating block-encodings

Rough overview (in words)

Given one or more block-encodings, we often want to form a single block-encoding of a product, tensor product, or linear combination of the individual block-encoded operators. This can be achieved as outlined below, using additional ancilla qubits.

Rough overview (in math) and resource cost

We will consider the case of two operators A and B, with straightforward generalizations to additional operators [431]. We are given an (α, a, ϵ_a)-block-encoding U_A of A, and a (β, b, ϵ_b)-block-encoding U_B of B. Operators A and B act on system qubits s.

Products: The operation $U_{AB} := (I_b \otimes U_A)(U_B \otimes I_a)$ is an $(\alpha\beta, a+b, \alpha\epsilon_b + \beta\epsilon_a)$-block-encoding of AB [431, Lemma 53], where I_x denotes the identity operator on x qubits (see Fig. 10.1). For example, if $a = b$, this construction uses twice as many ancilla qubits for block-encoding the product compared to the block-encoding of the individual matrices. In fact, we can assume without loss of generality that $a = b$ (by taking the maximum of the two) and improve the construction using the circuit in Fig. 10.2, which uses $a + 1$ ancilla qubits instead of $2a$. This idea has been generalized to encompass products of L block-encodings using only $a + \lceil \log_2(L) \rceil + 1$ ancillas (rather than aL), where it is known as the "compression gadget"; see [718, Lemma 13] and [379, Lemma 3].

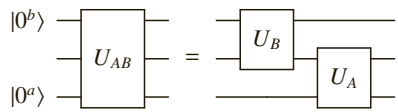

Figure 10.1 Implementing the block-encoding U_{AB} of AB that acts on s qubits. The cost is $a + b$ ancilla qubits, and one call to each of U_A, U_B.

Tensor products: The operation $U_{A\otimes B} := (U_A \otimes U_B)$ is an $(\alpha\beta, a+b, \alpha\epsilon_b + \beta\epsilon_a)$-block-encoding of the operator $A \otimes B$, as depicted in Fig. 10.3.

Linear combinations: Linear combinations of block-encodings can be viewed as a generalization of the linear combination of unitaries (LCU) trick [281]. We wish to implement a block-encoding of $\sum_{i=0}^{L-1} c_i A_i$, where

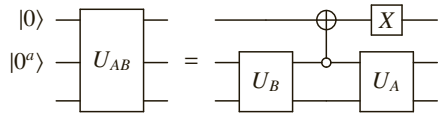

Figure 10.2 Implementing the block-encoding U_{AB} of AB for the case where both U_A and U_B act on a ancilla qubits. The controlled gate is an a-controlled generalized Toffoli gate.

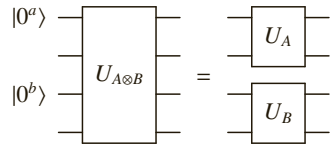

Figure 10.3 Implementing the block-encoding $U_{A\otimes B}$ of $A \otimes B$ that acts on $2s$ qubits. The cost is $a + b$ ancilla qubits, and one call to each of U_A, U_B.

$c_i \in \mathbb{R}$ (the LCU trick can also be extended to complex coefficients) and define $\lambda := \sum_{i=0}^{L-1} |c_i|$. We consider L block-encodings U_i that are $(1, m, \epsilon_i)$-block-encodings of A_i. We note that in cases where the block-encodings have different α_i or m_i values, the former can be absorbed into the c_i values and the latter can be taken as $m = \max_i m_i$.

We first define an operator PREPARE by the following action on $|0^{\lceil \log_2(L) \rceil}\rangle$

$$\text{PREPARE}|0^{\lceil \log_2(L) \rceil}\rangle = \frac{1}{\sqrt{\lambda}} \sum_j \sqrt{|c_j|}|j\rangle$$

that prepares a weighted superposition on an ancilla register, such that the amplitudes are proportional to the square roots of the absolute values of the desired coefficients. We also define[2]

$$\text{SELECT} = \sum_{j=0}^{L-1} |j\rangle\langle j| \otimes \text{sign}(c_j) U_j.$$

We then have the following result

$$\begin{aligned} &\left(\langle 0^{\lceil \log_2(L) \rceil}| \otimes I\right) \text{PREPARE}^\dagger \cdot \text{SELECT} \cdot \text{PREPARE}\left(|0^{\lceil \log_2(L) \rceil}\rangle \otimes I\right) \\ &= \frac{1}{\lambda} \sum_{i=0}^{L-1} c_i U_i \,, \end{aligned} \tag{10.2}$$

[2] To be precise, for $j \notin \{0, 1, \ldots, L-1\}$ we define $\text{sign}(c_j) U_j := I$.

that is, $U_{\mathrm{LC}} := \mathrm{PREPARE}^\dagger \cdot \mathrm{SELECT} \cdot \mathrm{PREPARE}$ is a $(\lambda, \lceil \log_2(L) \rceil, 0)$-block-encoding of the LCU $\sum_i c_i U_i$, as depicted in Fig. 10.4. This is the standard LCU trick [281], and it does not require U_i to be block-encodings (or we can view them as $(1, 0, 0)$-block-encodings of themselves). This technique can be used in Hamiltonian simulation, or to instantiate a block-encoding.

If, as specified above, U_i are block-encodings of $\tilde{A}_i$ (which approximate A_i), we also have the following result

$$\left\| \left(\sum_{i=0}^{L-1} c_i A_i \right) - \lambda \left(\langle 0^{m+\lceil \log_2(L) \rceil}| \otimes I \right) U_{\mathrm{LC}} \left(|0^{m+\lceil \log_2(L) \rceil}\rangle \otimes I \right) \right\| \leq \sum_{i=0}^{L-1} |c_i| \epsilon_i.$$

Hence, U_{LC} is a $(\lambda, \lceil \log_2(L) \rceil + m, \lambda \max_i \epsilon_i)$-block-encoding of $\sum_{i=0}^{L-1} c_i A_i$.

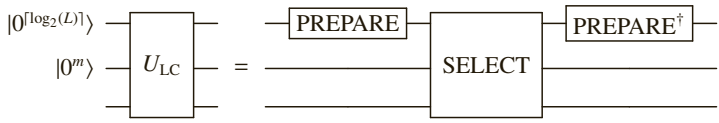

Figure 10.4 Implementing the block-encoding U_{LC} of $\sum_i c_i A_i$ that acts on s qubits. We require $\lceil \log_2(L) \rceil + m$ ancilla qubits. The regular LCU circuit is obtained by omitting the register $|0^m\rangle$ and the requirement that U_i are block-encodings. The gate complexity of PREPARE depends on the coefficients c_i but is $\Theta(L)$ in the worst case (using no additional ancilla qubits) [835]. We can also define PREPARE that leads to entanglement with a garbage register $\mathrm{PREPARE}|0^{\lceil \log_2(L) \rceil}\rangle|0^g\rangle = \lambda^{-0.5} \sum_i \sqrt{|c_i|}|i\rangle|G_i\rangle$, which can be seen to satisfy the relations required to implement the linear combination, Eq. (10.2). It can sometimes (e.g., [75]) be cheaper to implement this garbage-entangled PREPARE; see Section 17.2 on preparing states from classical data. The cost of SELECT depends on the form of U_i, but in the worst case requires $\Theta(L)$ primitive gates and $\Theta(L)$ calls to $|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U_i$ [283, 75], although this can be improved in some relevant special cases (e.g., [1011]). When U_i are multiqubit Pauli operators, [1084] provides a depth-optimized implementation of SELECT that achieves $O(\log(Ln))$ depth using $\Theta(Ln)$ total gates and total ancilla qubits.

Caveats

Performing linear algebraic manipulations of block-encodings using these primitives can quickly increase the ancilla count of the algorithm and worsen the normalization factor of the block-encoding. Amplifying a subnormalized block-encoding is possible, but costly, requiring an amount of time scaling roughly linearly in the amplification factor; see [715, 431]. Given a single block-encoded operator A, the above primitives can be used to implement a block-encoding of a polynomial in A. However, this can be achieved with much lower overhead using quantum singular value transformation (QSVT).

Example use cases

- Linear combination of block-encodings are used to obtain mixed-parity functions in QSVT required for Hamiltonian simulation.
- LCU trick is used for Hamiltonian simulation via Taylor series and to instantiate block-encodings of chemistry or condensed matter physics Hamiltonians (see, e.g., [75, 1011]).

Further reading

- References [428, Section 3.3] and [687, Section 7.3] contain a comprehensive discussion of manipulating block-encodings, including proofs of many of the results stated above.

10.3 Quantum signal processing

Rough overview (in words)

Quantum signal processing (QSP) [719] describes a method for nonlinear transformations of a signal parameter encoded in a single-qubit gate, using a structured sequence that interleaves the "signal gate" with fixed parameterized "modulation" gates. The technique was originally motivated by the desire to characterize pulse sequences used in nuclear magnetic resonance [719]. Remarkably, it has been shown [719, 477] that there is a rich family of polynomial transformations that are in one-to-one correspondence with appropriate modulation sequences; moreover, given such a polynomial, one can efficiently compute the corresponding modulation parameters.

Even more remarkably, this analysis holds not just for single-qubit "signal gates" but can be extended for multiqubit operators that *act* like single-qubit rotations when restricted to appropriate 2D subspaces [716]. This insight enables the implementation of block-encodings of polynomials of Hermitian or normal matrices when used in conjunction with qubitization. The two-step process of qubitization and QSP can be unified and generalized through quantum singular value transformation (QSVT).

Rough overview (in math)

We follow the "Wx convention" of QSP [431, 744]. We define the single-qubit signal operator

$$W(x) := \begin{pmatrix} x & \mathrm{i}\sqrt{1-x^2} \\ \mathrm{i}\sqrt{1-x^2} & x \end{pmatrix} = \mathrm{e}^{\mathrm{i}\arccos(x)X}$$

which is a single-qubit X rotation. We can verify that

$$W(x)^2 = \begin{pmatrix} 2x^2 - 1 & \cdot \\ \cdot & \cdot \end{pmatrix}$$
$$W(x)^3 = \begin{pmatrix} 4x^3 - 3x & \cdot \\ \cdot & \cdot \end{pmatrix}$$
$$\vdots$$
$$W(x)^n = \begin{pmatrix} T_n(x) & \cdot \\ \cdot & \cdot \end{pmatrix},$$

where $T_n(x)$ is the n-th Chebyshev polynomial of the first kind, showcasing that even a simple sequence of the signal unitaries can implement a rich family of polynomials of the signal x.

More complex behavior is obtained by interleaving $W(x)$ with parameterized single-qubit Z rotations $e^{i\phi_j Z}$. We define a QSP sequence as

$$U_{\mathrm{QSP}}(\Phi) := e^{i\phi_0 Z} \prod_{j=1}^{d} W(x) e^{i\phi_j Z},$$

where Φ denotes the vector of angles $(\phi_0, \phi_1, \ldots, \phi_d)$. The QSP sequence implements the following unitary

$$U_{\mathrm{QSP}}(\Phi) = \begin{pmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix}, \tag{10.3}$$

where $P(x), Q(x)$ are complex polynomials obeying a number of constraints (see below), and $P^*(x)$, $Q^*(x)$ denote their complex conjugates. Note also that the relationship between the sequence Φ and the corresponding polynomial $P(x)$ can be understood through nonlinear Fourier analysis [19].

Dominant resource cost (gates/qubits)

A QSP circuit that implements a degree-d polynomial in the signal parameter requires d uses of $W(x)$ and $d + 1$ fixed-angle Z rotations. There are efficient classical algorithms to determine the angles for a given target polynomial, either using high-precision arithmetic with $\sim d\log(d)$ bits of precision [477] (or more [431]—though this can be mitigated using heuristic techniques [255, 1065]) or in some regimes using more efficient optimization-based or iterative algorithms [356, 1018, 359, 360, 19, 18]. In particular, this line of work has culminated in [18] with an algorithm for finding the angles that is provably numerically stable, that is, it requires only polylog(d/ϵ) bits of precision to achieve ϵ error. An alternative method computes the angles directly

if both $P(x)$ and $Q(x)$ are known, and offers a way to compute $Q(x)$ if only $P(x)$ is known [782, 130]. Although these procedures are efficient in theory, in practice it may still be nontrivial to find the angles. Nevertheless, researchers reportedly computed angle sequences corresponding to various degree $d \approx 10^7$ polynomials [782, 130].

Caveats

As discussed above, not all polynomials can be implemented by a QSP sequence. Implementable polynomials must obey a number of constraints, which can be somewhat restrictive. For the standard QSP circuit $U_{\mathrm{QSP}}(\Phi)$ given above, the achievable polynomials pairs $P(x), Q(x) \in \mathbb{C}$ can be characterized by the following three conditions

- $\mathrm{Deg}(P) \leq d$ and $\mathrm{Deg}(Q) \leq d-1$,
- $\mathrm{Parity}(P) = \mathrm{Parity}(d)$ and $\mathrm{Parity}(Q) = \mathrm{Parity}(d-1)$,
- $\forall\ x \in [-1,1] : |P(x)|^2 + (1-x^2)|Q(x)|^2 = 1$ (required for Eq. (10.3) to be unitary).

This last requirement can be particularly limiting. A useful way to circumvent this for real functions is to encode the polynomial in the matrix element $\langle +|U_{\mathrm{QSP}}(\Phi)|+\rangle$ rather than in $\langle 0|U_{\mathrm{QSP}}(\Phi)|0\rangle$, where $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. This matrix element evaluates to

$$\langle +|U_{\mathrm{QSP}}(\Phi)|+\rangle = \mathrm{Re}[P(x)] + i\sqrt{1-x^2}\,\mathrm{Re}[Q(x)]\,.$$

Given a real target polynomial $f(x)$ with parity equal to $\mathrm{Parity}(d)$, we can guarantee that the matrix element evaluates to $f(x)$ by choosing $\mathrm{Re}[P(x)] = f(x)$ and $\mathrm{Re}[Q(x)] = 0$. The third condition above then reduces to $1 - f(x)^2 = |\mathrm{Im}[P(x)]|^2 + (1-x^2)|\mathrm{Im}[Q(x)]|^2$. By [431, Lemma 6], there exist choices for $\mathrm{Im}[P(x)]$ and $\mathrm{Im}[Q(x)]$ that satisfy this identity as well as the first two conditions above, provided $|f(x)| \leq 1\ \forall\ x \in [-1,1]$. In summary, we may implement any real polynomial $f(x)$ satisfying the requirements [431, Corollary 10]:

- $\mathrm{Deg}(f) = d$,
- $\mathrm{Parity}(f) = \mathrm{Parity}(d)$,
- $\forall\ x \in [-1,1] : |f(x)| \leq 1$.

There are several related conventions considered in the literature for the explicit form of the single-qubit operators used in QSP; a thorough discussion is given in [744, Appendix A]. One common form that links closely to qubitization and QSVT is the reflection convention, which replaces $W(x)$ by the

reflection

$$R(x) = \begin{pmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{pmatrix}, \tag{10.4}$$

and adjusts the parameters $\{\phi_j\}$ accordingly [431].

Example use cases

- Functions of Hermitian or normal matrices, in conjunction with qubitization, including for Hamiltonian simulation.
- Functions of general matrices via QSVT.
- Reference [357] applied QSP to beyond-Heisenberg-limit calibration of two-qubit gates in a superconducting system.

Further reading

- A pedagogical discussion of QSP [744].
- Detailed proofs of the key results of QSP [719, 431].
- Lecture notes on QSP [687, Section 7.6].

10.4 Qubitization

Rough overview (in words)

Qubitization is a method for using a block-encoding U_A of a Hermitian operator A to manipulate A, for example, implement A^2, or, more generally, some function $f(A)$ [717]. However, the eigenvalues of U_A are typically unrelated to those of A, and plain repeated applications of U_A do not in general produce the desired behavior. Qubitization converts the block-encoding U_A into a unitary operator W (sometimes called a qubiterate or a qubitized quantum walk operator) having the following guaranteed advantageous properties:

- W is a block-encoding of the operator A.
- The spectrum of W is directly related to the spectrum of A.
- Repeated application of W leads to structured behavior that can be cleanly analyzed.

This combination of features means that qubitization can be used for applying polynomial transformations to the spectrum of A. For example, repeated application of W implements Chebyshev polynomials of A, while other polynomials can also be implemented by using quantum signal processing [716, 717, 431].

The key observation is that a qubitization unitary W has eigenvalues and eigenvectors that relate in a nice way to those of A. Thus, one can also perform

quantum phase estimation on W to learn these quantities [841, 139], providing a potentially cheaper alternative to such tasks compared to approaches based on explicit Hamiltonian simulation for implementing $U = e^{iAt}$.

Rough overview (in math)

We are given a $(1, m, 0)$-block-encoding U_A of Hermitian A, such that

$$A = (\langle 0^m| \otimes I) U_A (|0^m\rangle \otimes I) \iff U_A = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix}.$$

First, we will assume U_A is also Hermitian (implying $U_A^2 = I$, where I is the identity matrix). Let A have spectral decomposition $A = \sum_\lambda \lambda |\lambda\rangle\langle\lambda|$. An application of U_A to an eigenstate $|\lambda\rangle$ of A gives

$$U_A|0^m\rangle|\lambda\rangle = \lambda|0^m\rangle|\lambda\rangle + \sqrt{1-\lambda^2}|\perp_{0^m,\lambda}\rangle,$$

where $|\perp_{0^m,\lambda}\rangle$ is a state perpendicular to $|0^m\rangle$.[3] Noting $U_A^2 = I$ reveals that the 2D subspace S_λ spanned by $\{|0^m\rangle|\lambda\rangle, |\perp_{0^m,\lambda}\rangle\}$ is invariant under the action of U_A. U_A restricted onto S_λ can be described by the matrix

$$\begin{array}{c} \\ |0^m\rangle|\lambda\rangle \\ |\perp_{0^m,\lambda}\rangle \end{array} \begin{array}{c} \begin{array}{cc} |0^m\rangle|\lambda\rangle & |\perp_{0^m,\lambda}\rangle \end{array} \\ \begin{pmatrix} \lambda & \sqrt{1-\lambda^2} \\ \sqrt{1-\lambda^2} & -\lambda \end{pmatrix} \end{array},$$

which is a 2D reflection with eigenvalues ± 1. Clearly, repeated application of (self-inverse) U_A can have limited effect on any input state. Qubitization uses a reflection $Z_{|0^m\rangle} = (2|0^m\rangle\langle 0^m| - I)$ to transform U_A into a Grover-like operator $W = Z_{|0^m\rangle} U_A$ which has the following matrix when restricted onto the invariant subspace S_λ in the $\{|0^m\rangle|\lambda\rangle, |\perp_{0^m,\lambda}\rangle\}$ basis

$$\begin{aligned} [W]_{\{|0^m\rangle|\lambda\rangle, |\perp_{0^m,\lambda}\rangle\}} &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \lambda & \sqrt{1-\lambda^2} \\ \sqrt{1-\lambda^2} & -\lambda \end{pmatrix} \\ &= \begin{pmatrix} \lambda & \sqrt{1-\lambda^2} \\ -\sqrt{1-\lambda^2} & \lambda \end{pmatrix}, \end{aligned}$$

showing that W is still a $(1, m, 0)$-block-encoding of A. This has the form of a Y-axis rotation

$$[W]_{\{|0^m\rangle|\lambda\rangle, |\perp_{0^m,\lambda}\rangle\}} = \begin{pmatrix} \cos(\theta_\lambda) & \sin(\theta_\lambda) \\ -\sin(\theta_\lambda) & \cos(\theta_\lambda) \end{pmatrix},$$

[3] If $\lambda = \pm 1$, then there is no need for $|\perp_{0^m,\lambda}\rangle$, and the subspace S_λ becomes 1D.

where $\theta_\lambda = \arccos(\lambda)$. Therefore, W has eigenvalues $e^{\pm i \arccos(\lambda)}$ with respective eigenvectors $(|0^m\rangle|\lambda\rangle \pm i|\perp_{0^m,\lambda}\rangle)/\sqrt{2}$, which can be accessed using quantum phase estimation.

Furthermore, we can see that on the span of the subspaces S_λ repeated application of W acts as

$$\begin{aligned} W^d &= \bigoplus_\lambda \begin{pmatrix} \cos(d\theta_\lambda) & \sin(d\theta_\lambda) \\ -\sin(d\theta_\lambda) & \cos(d\theta_\lambda) \end{pmatrix} \\ &= \bigoplus_\lambda \begin{pmatrix} T_d(\lambda) & \sqrt{1-\lambda^2}U_{d-1}(\lambda) \\ -\sqrt{1-\lambda^2}U_{d-1}(\lambda) & T_d(\lambda) \end{pmatrix} \\ &= \begin{pmatrix} T_d(A) & \cdot \\ \cdot & \cdot \end{pmatrix}, \end{aligned}$$

where $T_d(\cdot)$ and $U_d(\cdot)$ are degree-d Chebyshev polynomials of the first and second kind, respectively. Therefore, W^d applies the polynomial transformation T_d to each eigenvalue of A, thereby implementing $T_d(A)$.

Dominant resource cost (gates/qubits)

The resource cost of qubitization is inherited from the cost of the block-encoding. Given a Hermitian $(\alpha, m, 0)$-block-encoding U_A, the qubitization operator W is a (non-Hermitian) $(\alpha, m, 0)$-block-encoding, and it uses no additional qubits. The operation $Z_{|0^m\rangle} = (2|0^m\rangle\langle 0^m| - I)$ can be implemented (up to a global phase) with an m-qubit (anti)controlled $-Z$ gate, equivalent to an m-qubit Toffoli up to single-qubit gates. An example qubitization circuit is shown below in Fig. 10.5 for $m = 3$. Implementing a block-encoding of a degree-d Chebyshev polynomial applied to A requires d calls to U_A and $Z_{|0^m\rangle}$.

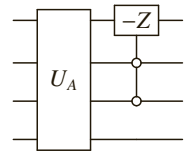

Figure 10.5 An example qubitization circuit using the Hermitian $(1, 3, 0)$-block-encoding U_A.

If the block-encoding U_A is not Hermitian, qubitization can be achieved using the construction of [717, Lemma 10] that uses one additional qubit, one call to controlled U_A, and one call to controlled $U_A^\dagger$ to implement the Hermitian

block-encoding

$$U'_A := ((HX) \otimes I)(|0\rangle\langle 0| \otimes U_A + |1\rangle\langle 1| \otimes U_A^\dagger)(H \otimes I)\,. \tag{10.5}$$

An alternative to qubitization is based on quantum singular value transformation (QSVT) that uses the sequence $Z_{|0^m\rangle} U_A^\dagger Z_{|0^m\rangle} U_A$, analogous to the earlier W^2, acting as

$$\begin{pmatrix} \lambda & \sqrt{1-\lambda^2} \\ -\sqrt{1-\lambda^2} & \lambda \end{pmatrix}^2$$

on a 2D subspace analogous to S_λ. The approach can be extended to odd-degree polynomials with a single additional application of $Z_{|0^m\rangle} U_A$ [431]. The advantage of this approach is that it does not require U_A to be Hermitian, thus there is no need for an additional qubit or calls to controlled $U_A^{\pm 1}$. This approach may be referred to as "quantum eigenvalue transformation" [687, 356] as this is a special case of QSVT applied to Hermitian A.

Caveats

The original formulation of qubitization [717] discussed above requires a Hermitian or normal block-encoded matrix A. The concept can be extended to general (non-square) matrices via QSVT, providing a significant generalization, however, in some cases quantum signal processing and its generalized versions [477, 255] can exploit additional structure that comes, for example, from the extra symmetries of Hermitian block-encodings, leading to potential constant-factor savings. Consider, for example, Hamiltonian simulation, where QSVT separately implements $\sin(tH)$ and $\cos(tH)$ using a block-encoding U_H of the Hamiltonian H, and applies a three-step oblivious amplification procedure on top of linear combination of unitaries to implement $\exp(itH)$ [431]. Meanwhile, quantum signal processing implements $\exp(itH)$ directly but requires an additional ancilla qubit and controlled access to a Hermitian block-encoding U'_H, which, when implemented via Eq. (10.5), uses both controlled U_H and $U_H^\dagger$, resulting in a factor of ~ 4 overhead. Altogether these considerations suggest that the QSVT-based approach might have a slightly better constant-factor overhead, particularly when controlled U_H is significantly more costly to implement than U_H. If U_H is already Hermitian, then quantum signal processing can have an improved complexity.

Example use cases

- Some quantum algorithms in quantum chemistry that compute energies perform phase estimation on a qubitization operator W implemented via calls to a block-encoding of the electronic structure Hamiltonian. This avoids the

approximation error incurred when performing phase estimation on e^{iHt}, implemented via Hamiltonian simulation [841, 139].
- Qubitization acts as a precursor to QSVT, which extends the concept to general matrices and unifies it with quantum signal processing.

Further reading

- Original introduction of qubitization [717] and QSVT [431].
- A pedagogical overview of quantum signal processing, its lifting to QSVT, and their applications [744].
- Reference [687, Chapters 7 & 8] provides an accessible derivation of qubitization and QSVT.

10.5 Quantum singular value transformation

Rough overview (in words)

Quantum singular value transformation (QSVT) can be viewed as both a unification and generalization of qubitization and quantum signal processing. Given a block-encoding U_A of a general matrix A, QSVT enables the transformation of the singular values of A by a polynomial $f(\cdot)$. In QSVT, there is a one-to-one correspondence between the desired polynomial transformation and its quantum circuit implementation whose parameters can be found by efficient classical algorithms.

It transpires that a number of existing quantum algorithms have simple and optimal (or near-optimal) implementations via the QSVT framework, including Hamiltonian simulation [716, 717, 431], amplitude amplification and estimation [431, 855], quantum linear systems solving [431, 744], Gibbs sampling [431], algorithms for topological data analysis [514, 755, 143], and quantum phase estimation [744, 854].

Rough overview (in math)

We are given a $(1, m, 0)$-block-encoding U_A of operator A (for simplicity, we will restrict our presentation to square matrices A, noting there is a straightforward generalization to non-square A [431]), such that

$$A = (\langle 0^m| \otimes I) U_A (|0^m\rangle \otimes I),$$

where $|0^m\rangle$ denotes $|0\rangle^{\otimes m}$. The matrix A has a singular value decomposition (SVD)

$$A = \sum_i \sigma_i |w_i\rangle\langle v_i|.$$

QSVT provides a method for implementing

$$f^{(SV)}(A) := \begin{cases} \sum_i f(\sigma_i)|w_i\rangle\langle v_i| & \text{if } f \text{ is odd, and} \\ \sum_i f(\sigma_i)|v_i\rangle\langle v_i| & \text{if } f \text{ is even,} \end{cases}$$

for certain definite-parity polynomials $f\colon [-1,1] \to \mathbb{C}$, such that $|f(x)| \leq 1 \; \forall \; x \in [-1,1]$. Crucially, QSVT does not require us to know the SVD in advance; the transformation is carried out automatically by following an SVD-agnostic procedure outlined below. Note that $f^{(SV)}(A)$ only coincides with the matrix function $f(A)$ for Hermitian A (see §Caveats, below). In the Hermitian case, we can also obtain block-encodings of mixed-parity or complex functions by taking linear combinations of block-encodings—see [356] for examples.

By considering $U_A|0^m\rangle|v_i\rangle$ and $U_A^\dagger|0^m\rangle|w_i\rangle$ one can show that (see [687] for a step-by-step derivation) U_A and $U_A^\dagger$ act as linear maps between the 2D subspaces $S_i := \text{Span}\{|0^m\rangle|v_i\rangle, |\perp_i\rangle\}$ and $S_i' := \text{Span}\{|0^m\rangle|w_i\rangle, |\perp_i'\rangle\}$, with U_A being a transition matrix between these bases given by

$$\begin{array}{c} \\ |0^m\rangle|w_i\rangle \\ |\perp_i'\rangle \end{array} \begin{array}{c} \begin{array}{cc} |0^m\rangle|v_i\rangle & |\perp_i\rangle \end{array} \\ \begin{pmatrix} \sigma_i & \sqrt{1-\sigma_i^2} \\ \sqrt{1-\sigma_i^2} & -\sigma_i \end{pmatrix} \end{array}, \tag{10.6}$$

where both $|\perp_i\rangle, |\perp_i'\rangle$ are orthogonal to $|0^m\rangle$ (but not necessarily to each other).[4] The 2D subspace S_i is invariant under the operation $W := Z_{|0^m\rangle} U_A^\dagger Z_{|0^m\rangle} U_A$ (where $Z_{|0^m\rangle} = (2|0^m\rangle\langle 0^m| - I)$). The operation W, restricted onto the 2D subspace S_i, is written as

$$\begin{pmatrix} \sigma_i & \sqrt{1-\sigma_i^2} \\ -\sqrt{1-\sigma_i^2} & \sigma_i \end{pmatrix}^2.$$

An additional application of $Z_{|0^m\rangle} U_A$ maps back into the S_i' subspace. By analogy with qubitization, repeated application of W applies a Chebyshev polynomial to each of the singular values of A. In analogy with quantum signal processing, by lifting the $Z_{|0^m\rangle}$ reflection operation to a (controlled) rotation $e^{i\phi_j Z_{|0^m\rangle}}$ we can impose polynomial transformations of the singular values of A, which then induce the claimed polynomial transformation of A. It is typically convenient to use an additional ancilla qubit to implement $e^{i\phi_j Z_{|0^m\rangle}}$.

[4] If $\sigma_i = 1$, then there is no need for $|\perp_i\rangle, |\perp_i'\rangle$, and the subspaces S_i, S_i' become 1D.

We define a QSVT circuit as the unitary sequence

$$U_\Phi := \begin{cases} e^{i\phi_1 Z_{|0^m\rangle}} U_A \prod_{j=1}^{(d-1)/2} \left(e^{i\phi_{2j} Z_{|0^m\rangle}} U_A^\dagger e^{i\phi_{2j+1} Z_{|0^m\rangle}} U_A \right) & \text{if } d \text{ is odd, and} \\ \prod_{j=1}^{d/2} \left(e^{i\phi_{2j-1} Z_{|0^m\rangle}} U_A^\dagger e^{i\phi_{2j} Z_{|0^m\rangle}} U_A \right) & \text{if } d \text{ is even,} \end{cases}$$

where $\Phi = (\phi_1, \phi_2, \ldots, \phi_d)$. We have that

$$(\langle 0^m| \otimes I) U_\Phi (|0^m\rangle \otimes I) = P^{(SV)}(A) = \begin{cases} \sum_i P(\sigma_i) |w_i\rangle\langle v_i|, & \text{for odd } d, \text{ and} \\ \sum_i P(\sigma_i) |v_i\rangle\langle v_i|, & \text{for even } d, \end{cases}$$

that is, the unitary U_Φ is a block-encoding of $P^{(SV)}(A)$, were P is the same polynomial that appears in quantum signal processing because the 2D matrix of Eq. (10.6) has the same form as the analogous 2D matrix in Eq. (10.4). We note that the constraints on the polynomials typically preclude direct implementation of the desired function as outlined above. By exploiting that $-\Phi$ implements P^*, we can use the circuit shown in Fig. 10.6 to implement a block-encoding of

$$P_{\Re}(A) = (\langle +| \otimes \langle 0^m| \otimes I)(|0\rangle\langle 0| \otimes U_\Phi + |1\rangle\langle 1| \otimes U_{-\Phi})(|+\rangle \otimes |0^m\rangle \otimes I)$$

for any definite-parity polynomial $P_{\Re} : [-1, 1] \to [-1, 1]$ by appropriately choosing Φ to implement a complex polynomial that fulfills the QSP conditions and then taking linear combinations of $U_\Phi, U_{-\Phi}$ to give a block-encoding of $P_{\Re}(A)$ [431, 744, 356].

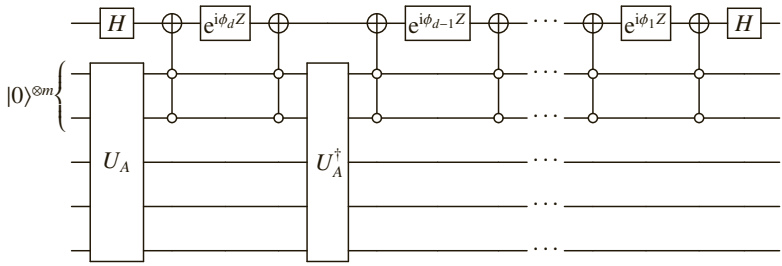

Figure 10.6 The QSVT circuit U_Φ which transforms a block-encoding U_A of A into a block-encoding of $f(A)$ for definite-parity $f : [-1, 1] \to [-1, 1]$ polynomial of degree d. As discussed in the main text, the angles $\{\phi_i\}$ can be calculated using efficient classical algorithms.

Dominant resource cost (gates/qubits)

Given a degree-d even-parity polynomial $f : [-1, 1] \to [-1, 1]$ and a $(1, m, 0)$-block-encoding U_A of A, one can implement a block-encoding of $f(A)$ using

$d/2$ calls to U_A, $d/2$ calls to $U_A^\dagger$, $2d$ m-controlled Toffoli gates, and d single-qubit Z rotations (as shown in Fig. 10.6). Implementing a degree $d+1$ odd polynomial additionally requires another call to U_A, another two m-controlled Toffoli gates, and another single-qubit Z rotation. The QSVT circuit implements a $(1, m+1, 0)$-block-encoding of $f(A)$.

If U_A is imperfect (i.e., it is a $(1, m, \epsilon)$-block-encoding of A), then [431, Lemma 22] shows that the error in $f(A)$ is bounded by $4d\sqrt{\epsilon}$; that is, QSVT implements a $(1, m+1, 4d\sqrt{\epsilon})$-block-encoding of $f(A)$. Moreover, if the norm of A is bounded away from 1, for example, $\|A\| \leq 1/2$, then the perturbation bound can be improved to $O(d\epsilon)$ [431, Lemma 23].

Given an initial state $|\psi\rangle$, the success probability of implementing $f(A)|\psi\rangle$ is given by $|\langle\psi|f(A)^\dagger f(A)|\psi\rangle|^2$.

Caveats

Since the output must be subnormalized to ensure the existence of a unitary block-encoding of $f(A)$, f must satisfy $|f(x)| \leq 1 \; \forall \; x \in [-1, 1]$.

As noted above, $f^{(SV)}(A)$ is only guaranteed to coincide with the matrix function $f(A)$ for Hermitian A. As an example, choosing $f(x) = x^2$ we have $f^{(SV)}(A) = \sum_i \sigma_i^2 |v_i\rangle\langle v_i| = A^\dagger A$ whereas $A^2 = \sum_{i,j} \sigma_i \sigma_j |w_i\rangle\langle v_i|w_j\rangle\langle v_j|$. As discussed above, for the Hermitian case we can implement a block-encoding of a mixed-parity function f by taking linear combinations of block-encodings of its even and odd parts. However, in the general case when $|w_i\rangle$ and $|v_i\rangle$ do not coincide, it does not seem to be possible to remove the parity constraint, as the odd $\sum_i f_{\text{odd}}(\sigma_i)|w_i\rangle\langle v_i|$ and even $\sum_i f_{\text{even}}(\sigma_i)|v_i\rangle\langle v_i|$ singular value transforms potentially map to different subspaces.

As discussed in Section 10.3 on quantum signal processing, computing the angle sequence Φ can be a nontrivial classical task. Several approaches for accomplishing this task have been studied [477, 431, 255, 1065, 356, 1018, 359, 360, 19, 18, 782, 130], and researchers have reported computing angle sequences for polynomials up to $d \approx 10^7$ [782, 130].

As noted above, if $f(A)$ has small singular values, then preparing a quantum state $f(A)|\psi\rangle$ might require many repeated uses of its block-encoding, thus the normalization factor of f plays a crucial role in efficiency.

In many applications, one seeks to apply a function that is not a polynomial (e.g., e^x, e^{ix}, $\text{erf}(x)$). In such cases, one needs to first approximate the desired function by a polynomial (incurring an approximation error ϵ) in order to apply QSVT.

Example use cases

- Linear equation solving: Apply a polynomial approximation of $1/x$ to a block-encoding of $A^\dagger$ to get an approximate block-encoding of the pseudoinverse A^+.
- Hamiltonian simulation: Apply polynomial approximations of $\sin(x)$ and $\cos(x)$ to a block-encoding of a Hamiltonian H and combine them with linear combination of unitaries and amplitude amplification to obtain a block-encoding of e^{iHt}.
- Fixed-point amplitude amplification [1067]: Construct a polynomial that maps values in the domain $[a_{\min}, 1]$ to the range $[1-\delta, 1]$, and apply this polynomial to a state-preparation unitary that prepares the desired state with amplitude a. The result is amplification of the amplitude to at least $1-\delta$ as long as $a > a_{\min}$.
- For additional applications, see [431, 854, 744, 689, 688].

Further reading

- The QSVT framework was introduced in [431] and is also discussed in detail in [428].
- A pedagogical tutorial of the QSVT framework is given in [744, 687].
- A streamlined derivation of QSVT is presented in [980].