Quantum Linear Algebra

Nihar Shah

May 19, 2025

Quantum Linear Algebra

Nihar Shah

Department of Computational and Data Science
Indian Institute of Science

Abstract

At a high level of abstraction, quantum computers compose unitary matrices, and do so with classically unparalleled efficiency. This hints at quantum speedup 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, and take their sums or products, or implement more general matrix functions. these tasks are collectively referred to as "quantum linear algebra", the building blocks are discussed in this section.

The techniques described in this section 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 in quantum circuits, inspired by approaches based on purification, dilation (that is, representing an incoherent state or operation as a coherent one with the help of an ancillary system for example, Stinespring representation or Stinespring dilation), and post-selection. Quantum signal processing (QSP) was discovered as a by product of the characterization of single single-qubit pulse sequences used in nuclear magnetic resonance, 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 combination of unitaries techniques based on matrix exponentials implemented by Hamiltonian simulation, or Chebyshev polynomials of operators implemented via quantum walk techniques. Such matrix exponentials or Chebyshev polynomials can be implemented, e.g, via qubitization of a block-encoded operator. In parallel to progress on advanced amplitude amplification techniques, it was recognized that QSP can be "lifted" for applying polynomial transformation to the eigenvalues of the 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, providing generalizations and more efficient implementations of a number of existing quantum algorithms and leading to the discovery of several new algorithms.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Phani Motamarri, for the continuous support of my study and related, for his patience, and immense knowledge. His guidance helped me through the time of research and writing of this thesis.

Besides my advisor, I am also grateful to the faculty members and staff at the Department of Computational and Data Science of Indian Institute of Science for their assistance and support. Special thanks to MATRIX lab for providing a stimulating and collaborative research environment.

Last but not the least, I would like to thank my family: my parents, for supporting me spiritually throughout writing this thesis and my life in general.

Contents

Chapter 1

Block Encoding

1.1 Motivation

In this section we describe a generic toolbox for implementing matrix calculations on a quantum computer in an operational way, representing the vectors as quantum states. The matrix arithmetic methodology proposed carries out all calculations in an operational way , such that the matrices are represented by blocks of unitary operators of the quantum system, thereby can in principle result in exponential speedups-in terms of the dimension of the matrices.

Here the results are in an intuitively structured way. First we define how to represent arbitrary matrices as blocks of unitaries, and show how to efficiently encode various matrices this way.

In quantum algorithms, the quantum gates that are applied to quantum states are necessarily unitary operations. 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 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 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, 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 $\mathcal{O}(1/\epsilon)$ uses of the block-encoding unitary in general (by using 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 of 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-values measures (POVMs), Gram matrices, sparse-access 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 from Gram matrices can be instantiated using state-preparation and quantum random access memory (QRAM) as subroutines.

1.2 Introduction

In order to perform matrix computations, we must first address the problem of the input model: how to get access to information in matrix $A \in \mathbb{C}^{N\times N}(N = 2^n)$ which is generally a non-unitary matrix, into the quantum computer? One possible input model is given via the unitary $e^{\iota\tau A}$ (if A is not Hermitian, in some scenarios we an consider its Hermitian version via the dilation method).

Many standard Linear algebra problems can be solved on a quantum computer using recently developed quantum linear algebra problems that make use of block encodings and quantum eigenvalue/ singular value transformations.

A block encoding embeds a properly scaled matrix of interest A in a larger Unitary transformation U that can be decomposed into a product of simpler unitaries and implemented efficiently on a quantum computer. Although, quantum algorithms can potentially achieve exponential speedup in solving linear algebra problems compared to the best classical algorithm, such gain in efficiency ultimately hinges on our ability to construct an efficient quantum circuit for the block encoding of A, which is difficult in general, and not trivial even for well structured sparse matrices. In this chapter, we talk about how efficient quantum circuits can be explicitly construct for some well

structured spare matrices and discuss a few examples and strategies used in these constructions.

In recent years, a new class of quantum algorithms have been developed to solve standard linear algebra problems on quantum computers. These algorithms use the technique of block encoding, to embed a properly scaled matrix A of interest in a larger unitary matrix U_A to a carefully prepared initial state and performing measurements on a subset of qubits. furthermore, if A is Hermitian, by using the technique of the quantum eigenvalue transformation, one can block encode a certain matrix polynomial $p(A)$ by another unitary $U_{p(A)}$ efficiently using U_A as the building block. This procedure can be generalized to non-Hermitian matrices A using a technique called singular value transformation. The larger unitary $U_{p(A)}$ can be decomposed onto a product of simpler unitaries consisting of 2×2 single qubit unitaries, multi-qubit controlled-NOTs, U_A and its Hermitian conjugate. Because approximate solutions to many large-scale linear algebra problems such as linear system of equations, least squares problems and eigenvalue problems (produced by iterative methods) can often be expressed as $p(A)v_0$ for some initial vector v_0 for some initial vector v_0, the possibility to block encode $p(A)$ will enable us to solve these problems on a quantum computer that perform unitary transformations.

In order to implement this quantum linear algebra algorithms, we need to further express the block encoding matrix U_A as a product of simpler unitaries i.e. we need ot express U_A as an efficient quantum circuit. Although it is suggested that such a quantum circuit can be constructed in theory for certain sparse matrices, the proposed construction relies on the availability of "oracle" that can efficiently block encode both the nonzero structure of A and numerical values of the nonzero matrix elements. However, for a general sparse matrix A, finding these "oracles" is entirely non-trivial. Even for spare matrices that have a well defined non-zero structure and a small set of nonzero matrix elements, encoding the structure and matrix elements by an efficient quantum circuit is not an easy task.

In this chapter, we provide a few accessible examples on how to explicitly construct efficient quantum circuits for some well structured spare matrices. In particular, some general strategies for writing down the oracles for the non-zero structure and non-zero matrix elements of those matrices. In addition we will also provide circuit diagrams and how these circuits can be easily constructed using Python Qiskit.

The block encoding of A is not unique. We will show a few different encoding schemes and the corresponding quantum circuits. For a sparse matrix A that has at most s nonzero elements per column, which we refer to as an s-sparse matrix, the general strategies we use to construct a quantum circuit actually block encodes A/s. For many applications, the extra scaling factor does not introduce significant issues.

1.3 Query model for matrix entries

The query model for sparse matrices is based on certain quantum oracles. In some scenarios, these quantum oracles can be implemented all the way to the elementary gate level. Throughout the discussion we assume A is an n-qubit, square matrix, and

$$\|A\|_{max} = \max_{ij} |A_{ij}| < 1$$

If the $\|A\|_{max} \geq 1$, we can simply consider the rescaled matrix $\tilde{A}/\alpha$ for some $\alpha > \|A\|_{max}$. To query the entries of a matrix, the desired oracle takes the following general form

$$O_A |0\rangle |i\rangle |j\rangle = \left(A_{ij} |0\rangle + \sqrt{1 - |A_{ij}|^2} |1\rangle \right) |i\rangle |j\rangle$$

In other words, given $i, j \in [N]$ and a signal bit 0, O_A performs a controlled rotation (controlling on i,j) of the signal bit, which encodes the information in terms of amplitude of $|0\rangle$.

However, the classical information in A is usually not stored natively in terms of such an oracle O_A. Sometimes it is more natural to assume that there is an oracle

$$\tilde{O}_A |0^{d'}\rangle |i\rangle |j\rangle = |\tilde{A}_{ij}\rangle |i\rangle |j\rangle$$

where $\tilde{A}_{ij}$ is a d'-bit fixed point representation of A_{ij}, and the value of $\tilde{A}_{ij}$ is either computed on the fly with a quantum computer, or obtained through an external database. In either case, the implementation of $\tilde{O}_A$ may be challenging, and we will only consider the query complexity with respect to this oracle.

Using classical arithmetic operations, we can convert this oracle into an oracle

$$O'_A |0^d\rangle |i\rangle |j\rangle = |\tilde{\theta}_{ij}\rangle |i\rangle |j\rangle$$

where $0 \leq \tilde{\theta}_{ij} < 1$, and $\tilde{\theta}_{ij}$ is a d-bit representation of $\theta_{ij} = arccos(A_{ij})/\pi$. This step may require some additional work registers not shown here.

Now using the controlled rotation, the information of $\tilde{A}_{ij}$, $\tilde{\theta}_{ij}$ has now been transferred to the phase of the signal bit. We should then perform uncomputation and free the work work register storing such intermediate information $\tilde{A}_{ij}$, $\tilde{\theta}_{ij}$. The procedure is as follows:

$$\begin{aligned}
|0\rangle |0^d\rangle |i\rangle |j\rangle &\xrightarrow{O'_A} |0\rangle |\tilde{\theta}_{ij}\rangle |i\rangle |j\rangle \\
&\xrightarrow{CR} \left(A_{ij} |0\rangle + \sqrt{1 - |A_{ij}|^2} |1\rangle \right) |\tilde{\theta}_{ij}\rangle |i\rangle |j\rangle \\
&\xrightarrow{(O'_A)^{-1}} \left(A_{ij} |0\rangle + \sqrt{1 - |A_{ij}|^2 x} |1\rangle \right) ket0^d |i\rangle |j\rangle
\end{aligned}$$

From now on, we will always assume that the matrix entries of A can be queried using the phase oracle O_A or its variants.

1.4 Block Encoding

Block encoding is a technique for embedding a properly scaled nonunitary matrix $A \in \mathbb{C}^{N\times N}$ into a unitary matrix U_A. **Note that a properly scaled A, we mean that A is scaled to satisfy $\|A\|_2 = \sigma_{max} \leq 1$. Without such a scaling, a block encoding of A may not exist because the singular values of any submatrix blocks of a unitary matrix must be bounded by** 1. Furthermore, if there are some constraints on the type of unitary it can construct e.g. the type of quantum gates available on a quantum computer, we may not be able to find U_A that block encodes A exactly. In this chapter, we assume that A has already been properly scaled, and there is no constraints on the quantum gates we can use to construct a quantum circuit representation of the unitary matrix U_A. The simplest example of block encoding is the following: assume we can find a (n+1)qubit unitary U (i.e. $U \in \mathbb{C}^{2N\times 2N}$ such that

$$U_A = \begin{bmatrix} A & * \\ * & * \end{bmatrix}$$

here * means that the corresponding entries are irrelevant (yet to be determined), then for any n-qubit quantum state $|b\rangle$, we can consider the state

$$|0, b\rangle = |0\rangle |b\rangle = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

and

$$U_A |0, b\rangle = \begin{bmatrix} Ab \\ * \end{bmatrix} = |0\rangle A |b\rangle + |1\rangle |*\rangle$$

Here the unnormalized state $|\perp\rangle = |1\rangle |*\rangle$ can be written as $|1\rangle |\psi\rangle$ for some unnormalized state $|\psi\rangle$, that is irrelevant to the computation of $A|b\rangle$. In particular, it satisfies the orthogonality relation.

$$(\langle 0| \otimes I_n) |\perp\rangle = 0$$

in order to obtain $A|b\rangle$, we need to ensure measure the qubit 0 and only keep the state if it returns 0. This can be summarised into the following quantum circuit as shown in the figure 1.1. Note that the output state is normalized after the measurement

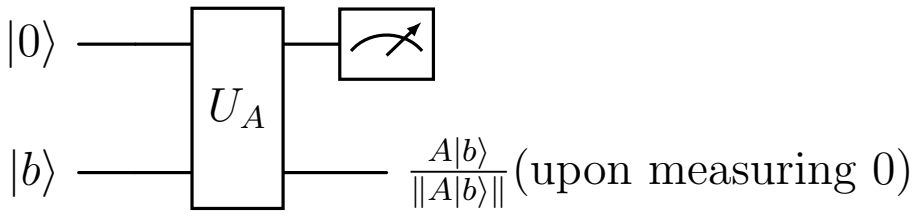


Figure 1.1: Circuit for Block Encoding A using one Ancilla qubit

takes place. The success probability of obtaining 0 from the measurement can be computed as

$$p(0) = \|A\,|b\rangle\,\|^2 = \langle b|A^\dagger A\,|b\rangle\rangle$$

So the missing information of norm $\|A\,|b\rangle\,\|$ can be recovered via the success probability $p(0)$ if needed. We find that the success probability is only determined by $A\,|b\rangle$, and is independent of other irrelevant components of U_A.

Note that we may not need to restrict the matrix U_A to be a $(n+1)$-qubit matrix. if we can find any (n+m)-qubit matrix U_A so that

$$U_A = \begin{bmatrix} A & * & \dots & * \\ * & * & \dots & * \\ \vdots & \vdots & \vdots & \vdots \\ * & * & \dots & * \end{bmatrix}$$

Here each $*$ stands for an n-qubit matrix, and there are 2^m block rows/columns in U_A. The relation above can be written compactly using the braket notation as

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

since $\langle 0^m| \otimes I_n$ will be a block row vector with Identity matrix (of size $2^n \times 2^n$) at the first entry and 0 everywhere else. Similarly, $|0^m\rangle \otimes I_n$ will be a block column vector with Identity matrix at the first entry and 0 everywhere else. Thus, the above expression can be thought of as being simplified to

$$\langle 0^m| \otimes I_n = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & 1 \end{bmatrix} = \begin{bmatrix} I_n & 0 & 0 & \dots & 0 \end{bmatrix}$$

where I_n is the n-qubit identity matrix of size $2^n \times 2^n$ and 0 indicates block matrices of size $2^n \times 2^n$ with all entries 0. Thus, $\langle 0^m| \otimes I_n$ is a matrix of size $2^n \times 2^{n+m}$. Similarly for $|0^m\rangle \otimes I_n$, we get,

$$|0^m\rangle \otimes I_n = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & 1 \end{bmatrix} = \begin{bmatrix} I_n \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Thus, on putting it all together we get,

$$(\langle 0^m| \otimes I_n) U_A (|0^m\rangle \otimes I_n) = \begin{bmatrix} I_n & 0 & 0 & \dots & 0 \end{bmatrix} U_A \begin{bmatrix} I_n \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = A$$

The above expression can be thought of as simply picking the first n columns of U_A and then the first n rows of A. A necessary condition for the existence of U_A is $\|A\| \leq 1$. (Note: $\|A\|_{max} \leq 1$ does not guarantee that $\|A\| \leq 1$). However, if we can find sufficiently large α and U_A so that

$$A/\alpha = (\langle 0^m| \otimes I_n) U_A (|0^m\rangle \otimes I_n)$$

Measuring the m ancilla qubits and given that all m-qubits return 0, we still obtain the normalized state $\frac{A|b\rangle}{\|A|b\rangle\|}$. The number α is hidden in the success probability:

$$p(0^m) = \frac{1}{\alpha^2} \| A |b\rangle \|^2 = \frac{1}{\alpha^2} \langle b| A^\dagger A |b\rangle \rangle$$

So if α is chosen to be too large, the probability of obtaining all 0s from the measurement can be vanishingly small.

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:

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

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

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

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. This can be shown using two facts that since $(\langle 0|^{\otimes a} \otimes I)U_A(|0\rangle^{\otimes a} \otimes I) \leq 1$ since it is an embedding in the top-left block of unitary U_A and using the property of norms that $\|A + B\| \leq \|A\| + \|B\|$. Given,

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

Let $\widetilde{A} = (\langle 0|^{\otimes a} \otimes I)U_A(|0\rangle^{\otimes a} \otimes I)$ Dividing by positive real scalar α on both sides of the inequality we get,

$$\|\frac{A}{\alpha} - \frac{\widetilde{A}}{\alpha}\| \leq \frac{\epsilon}{\alpha}$$

Now, using the inequality $\|\frac{\widetilde{A}}{\alpha}\| \leq 1$, and adding this inequality on both the sides we get,

$$\|\frac{\widetilde{A}}{\alpha}\| + \|\frac{A}{\alpha} - \frac{\widetilde{A}}{\alpha}\| \leq 1 + \frac{\epsilon}{\alpha}$$

Now, using the property that $\|A + B\| \leq \|A\| + \|B\|$, we get,

$$\begin{aligned} \|\frac{\widetilde{A}}{\alpha} + \frac{A}{\alpha} - \frac{\widetilde{A}}{\alpha}\|\| \leq \|\frac{\widetilde{A}}{\alpha}\| + \|\frac{A}{\alpha} - \frac{\widetilde{A}}{\alpha}\| &\leq 1 + \frac{\epsilon}{\alpha} \\ \implies \|\frac{A}{\alpha}\| &\leq 1 + \frac{\epsilon}{\alpha} \\ \implies \|A\| &\leq \alpha + \epsilon \end{aligned}$$

Upon rearranging this gives us, $\alpha \geq \|A\| - \epsilon$. and 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 $|0\rangle^{\otimes a}|\psi\rangle$ and post-selecting on the first register being in the state $|0\rangle^{\otimes a}$ gives an output state proportional to $A|\psi\rangle$ in the second register.

Example 1.4.1. To give an example of block encoding, let us consider a 1×1 matrix $A = \alpha$, where $0 < \alpha < 1$. In this extremely simple case, a block encoding of A can be constructed as

$$U_A = \begin{bmatrix} \alpha & \sqrt{1-\alpha^2} \\ \sqrt{1-\alpha^2} & -\alpha \end{bmatrix}$$

or

$$U_A = \begin{bmatrix} \alpha & -\sqrt{1\alpha^2} \\ \sqrt{1-\alpha^2} & \alpha \end{bmatrix}$$

Although this type of block encoding can be extended to a properly scaled matrix A of a larger dimension to yield

$$U_A = \begin{bmatrix} A & (I - A^\dagger A)^{1/2} \\ (I - A^\dagger A)^{1/2} & -A \end{bmatrix}$$

or

$$\begin{bmatrix} A & -(I - A^\dagger A)^{1/2} \\ (I - A^\dagger A)^{1/2} & A \end{bmatrix}$$

this approach is not practical because it requires computing the square root of $A^\dagger A$, which requires computing and diagonalizing $A^\dagger A$. In general, there is no efficient algorithm to perform these operations on a quantum computer using $\mathcal{O}(poly(n))$ quantum gates.

Example 1.4.2. Let H be a Hamiltonian made up of m Pauli terms, meaning that

$$H = \sum_{a=1}^{m} \lambda_a E_a \text{ where } E_a \text{ is a tensor product of Pauli matrices}$$

Find an algorithm implementing a Unitary close to $e^{-\iota Ht}$, so that $\|U - e^{\iota Ht}\| \leq \epsilon$. Original solutions proceeded by using Trotter approximations: for r large enough,

$$e^{-\iota Ht} \approx (e^{-\iota E_1 t/r} e^{-\iota E_2 t/r} \dots e^{-\iota E_m t/r})^r$$

However, this solution is far from optimal, notably because implementing this approximation, requires $poly(1/\epsilon)$ gate complexity. Improved algorithms eventually developed into the framework below presented. This framework proceeds by:

1. Defining a type of quantum circuit called a "block-encoding"
2. Showing that, given λ_a and E_a, we can construct an efficient block-encoding of H.

3. Showing that we can get a block-encoding of (an approximation of) $e^{-\iota Ht}$ with few uses of the block-encoding to H.

4. Using this block-encoding to apply our approximation to a state.

A more general input model, as will be discussed in this chapter is called "block encoding". Of course, if A is a dense matrix without obvious structures, any input model will be very expensive (e.g. exponential in n) to implement. Therefore a commonly assumed input model is s-sparse there are at most s nonzero entries in each row/column of the matrix. Furthermore, we have an efficient procedure to get access to the location, as well as the value of the nonzero entries. This in general can again be a difficult task given that the number of nonzero entries can still be exponential in n for a sparse matrix. Some dense matrices may also be efficiently block encoded on quantum computers. This chapter will illustrate the block encoding procedure via a number of detailed examples.

Example 1.4.3. A more practical scheme that does not require computing the square root of A can be illustrated by the following real symmetric 2×2 example. Let

$$A = \begin{bmatrix} \alpha_1 & \alpha_2 \\ \alpha_2 & \alpha_1 \end{bmatrix}$$

where $|\alpha_1|, |\alpha_2| \leq 1$. It can be verified that the matrix

$$U_A = \frac{1}{2} \begin{bmatrix} U_\alpha & -U_\beta \\ U_\beta & U_\alpha \end{bmatrix}$$

is a block encoding of $A/2$, where

$$U_\alpha = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_1 & -\alpha_2 \\ \alpha_2 & \alpha_1 & -\alpha_2 & \alpha_1 \\ \alpha_1 & -\alpha_2 & \alpha_1 & \alpha_2 \\ -\alpha_2 & \alpha_1 & \alpha_2 & \alpha_2 \end{bmatrix}$$

and

$$U_\beta = \begin{bmatrix} \beta_1 & \beta_2 & \beta_1 & -\beta_2 \\ \beta_2 & \beta_1 & -\beta_2 & \beta_1 \\ \beta_1 & -\beta_2 & \beta_1 & \beta_2 \\ \beta_1 & -\beta_2 & \beta_1 & \beta_2 \\ -\beta_2 & \beta_1 & \beta_2 & \beta_1 \end{bmatrix}$$

with $\beta_1 = \sqrt{1-\alpha_1^2}$ and $\beta_2 = \sqrt{1-\alpha_2^2}$. To implement this unitary on a quantum computer, we must further decompose U_A as a product of simpler unitaries and

construct a quantum circuit with a limited number of quantum gates. This method of block encoding cannot be easily generalized to matrices of larger dimensions. However, for special matrices with special structures ,e.g. sparse matrices, it is possible to develop some general block encoding strategies which we will discuss in the next section.

Finally, it can be difficult to find U_A to encode A exactly. This is not a problem, since it is sufficient if we can find U_A to block encode A upto some error ϵ. We are now ready to give the definition of block encoding.

1.4.1 The Definition

We introduce a definition of block encoding which we are going to work with. The main idea is rto represent a subnormalized matrix as the upper-left block of a unitary.

$$U = \begin{bmatrix} A/\alpha & * \\ * & * \end{bmatrix} \implies A = \alpha(\langle 0| \otimes I)U(|0\rangle \otimes I)$$

Definition 1.4.1. (Block encoding) Given an n-qubit matrix A ($N = 2^n$), if we can find $\alpha, \epsilon \in \mathbb{R}_+$, and an $(m+n)$-qubit unitary matrix U_A so that

$$\|A - \alpha(\langle 0^m| \otimes I_n)U_A(|0^m\rangle \otimes I_n)\| \leq \epsilon$$

then U_A is called an (α, m, ϵ)-block encoding of A. When the block encoding is exact with $\epsilon = 0$, U_A is called an (α, m)-block encoding of A. The set of all (α, m, ϵ)-block-encoding of A is denoted by $BE_{\alpha,m}(A, \epsilon)$, and we define $BE_{\alpha,m}A = BE(A, 0)$.

Here, m is the number of ancilla bits used to block encode A.

Important Note

In addition to the definition of block-encoding, one can also define an asymmetric version as follows:

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

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.

Assume we know each matrix element of the n-qubit matrix A_{ij}, and we are given an (n+m)-qubit unitary U_A. In order to verify that $U_A \in BE_{1,m}(A)$ we only need to verify that

$$\langle 0^m| \langle i| U_A |0^m\rangle |j\rangle = A_{ij}$$

Here, $|0^m\rangle |j\rangle$ will be a column vector of size $(2^m \times 1 \otimes 2^n \times 1 = 2^{m+n} \times 1)$ which will be 1 in the jth row and 0 everywhere else. Similar arguments can be used for establishing $\langle 0^m| \langle i|$ which will be a row vector with 1 in the ith column entry and 0 everywhere else. Thus, upon multiplying the matrices, the $|0^m\rangle |j\rangle$ will pick the jth column of the matrix U_A which will correspond to the jth column of the matrix A and $\langle 0^m| \langle i|$ will pick the ith row of U_A which will correspond to the ith row of A, thus, it will pick the i, j th entry of the matrix U_A which will be A_{ij}. U_A applied to the vector $|0^m, b\rangle$ can be obtained via the superposition principle.

Therefore we may first evaluate the state $U_A |0^m, j\rangle$, and perform inner product with $|0^m, i\rangle$ and verify the resulting the inner product is A_{ij}. We will also use the following technique frequently. Assume $U_A = U_B U_C$, and then

$$\langle 0^m, i|U_A |0^m, j\rangle\rangle = \langle 0^m, i|U_B U_C |0^m, j\rangle\rangle = (U_B^\dagger |0^m, i\rangle)^\dagger (U_C |0^m, j\rangle)$$

So we can evaluate the states $U_B^\dagger |0^m, i\rangle, U_C |0^m j\rangle$ independently, and then verify the inner product is A_{ij}. Such a calculation amounts to running the circuit shown in fig 1.2, and if the ancilla qubits are measured to be 0^m, the system qubits return the normalized state $\sum_i A_{ij} |i\rangle / \| \sum_i A_{ij} |i\rangle \|$.

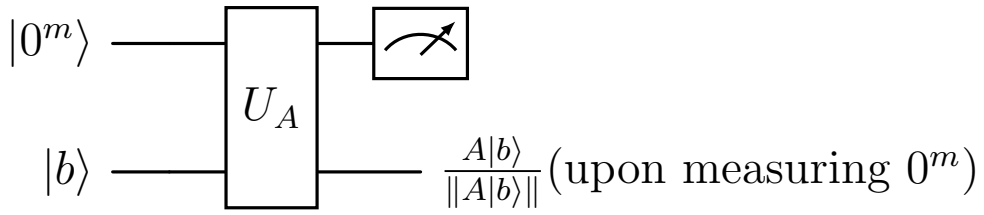


Figure 1.2: Circuit for general block encoding of A

Example 1.4.4. ((1,1)-block-encoding in general). For any n-qubit matrix A with $\|A\|_2 \leq 1$, the singular value decomposition (SVD) of A is denoted by $W\Sigma V^\dagger$, where all singular values in the diagonal matrix Σ belong to $[0, 1]$. Then we may construct

an (n+1)-qubit unitary matrix

$$U_A = \begin{bmatrix} W & 0 \\ 0 & I_n \end{bmatrix} \begin{bmatrix} \Sigma & \sqrt{I_n - \Sigma^2} \\ \sqrt{I_n - \Sigma^2} & -\Sigma \end{bmatrix} \begin{bmatrix} V^\dagger & 0 \\ 0 & I_n \end{bmatrix}$$
$$= \begin{bmatrix} A & W\sqrt{I_n - \Sigma^2} \\ \sqrt{I_n - \Sigma^2}V^\dagger & -\Sigma \end{bmatrix}$$

which is a $(1, 1)$-block-encoding of A.

Example 1.4.5. (Random circuit block encoded matrix). In some scenarios, we may want to construct a psuedo-random non-unitary matrix on quantum computers. Note that it would be highly inefficient if we first generate a dense psuedo-random matrix A classically and then feed it into the quantum computer using e.g. quantum-random-access memory (QRAM). Instead we would like to work with matrices that are inherently easy to generate on quantum computers. This inspires the random circuit based block encoding matrix (RACBEM) model. Instead of first identifying A and then finding its block encoding U_A, we reverse this thought process: we first identify a unitary U_A that is easy to implement on a quantum computer, and then ask which matrix can be block encoded by U_A.

This example shows that in principle, any matrix A with $\|A\|_2 \leq 1$ can be accessed via a $(1, 1, 0) - block - encoding$. In other words, A can be block encoded by an $(n+1)$-qubit random unitary U_A, and U_A can be constructed using only one-qubit unitaries and CNOT gates. The layout of the two-qubit operations can be designed to be compatible with the coupling map of the hardware.

Example 1.4.6. (Block encoding of a diagonal matrix). As a special case, let us consider the block encoding of a diagonal matrix. Since the row and column indices are the same, we may simplify the oracle into

$$O_A |0\rangle |i\rangle = \left(A_{ii} |0\rangle + \sqrt{1 - |A_{ii}|^2} |1\rangle \right) |i\rangle$$

In the case when the oracle $\tilde{O}_A$ is used, we may assume accordingly

$$\tilde{O}_A |0^{d'}\rangle |i\rangle = |\tilde{A}_{ii}\rangle |i\rangle$$

Let $U_A = O_A$. Direct calculation shows that for any $i, j \in [N]$,

$$\langle 0| \langle i|U_A|0\rangle |j\rangle = A_{ii}\delta_{ij}$$

This proves that $U_A \in BE_{1,1}(A)$i.e., U_A is a $(1, 1)$-block-encoding of the diagonal matrix A.

There are several ways of implementing block-encodings based on the choice of matrix. In particular, we will now describe how to construct block-encodings of unitary matrices, density operators, POVM operators, sparse-access matrices, and matrices stored in a QROM (by QROM we mean quantum read-only memory, which stores classical data that can be accessed in superposition). Quantum matrix arithmetics carries out all calculations in an operational way, meaning that the matrices are represented by block-encodings in principle enabling exponential speedups in terms of the dimension of the matrices.

Note that in all these cases since $\|U\| = 1$ we necessarily have $\|A\| \leq \alpha + \epsilon$. Also note that using the above defintiion it seems that we can only represent square matrices of size $2^n \times 2^n$. However, this is not really a restriction. Suppose that $A \in \mathbb{C}^{n\times m}$, where $n, m \leq 2^s$. Then we can define an embedding matrices denoted by $A_e \in \mathbb{C}^{2^s\times 2^s}$ such that the top-left block of A_e is A and all other elements are zero. This embedding is a faithful representation of the matrices. Suppose that $A, B \in \mathbb{C}^{n\times m}$ are matrices, then $A_e + B_e = (A + B)_e$. Moreover, suppose $X \in \mathbb{C}^{m\times k}$ for some $k \leq 2^s$, then $A_e \cdot C_e = (A \cdot C)_e$.

1.5 Unitary Matrices - Trivial Block Encoding

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.

Definition 1.5.1. (Trivial block-encoding) A unitary matrix is a $(1, 0, 0)$-block encoding of itself.

If we ϵ-approximately implement a unitary U using a ancilla qubits via a unitary $\tilde{U}$ acting jointly on the system and the ancilla qubits, then $\tilde{U}$ is an $(1, a, \epsilon)$-block encoding of U. This is also a rather trivial encoding. Note that we make a slight distinction between ancilla qubits that are exactly returned to their original state after the computation and the ones that might pick up some error. The latter qubits we will treat as part of the encoding, and the former qubits we usually treat as purely ancillary qubits.

Now we present some non-trivial ways for constructing block-encodings, which will serve as a toolbox for efficiently inputting and representing matrices for arithmetic computations on a quantum computer. We will denote by I_w a w-qubit identity operator, and let $SWAP_w$ denote the swap operation of two w qubit register. We denote by CNOT the controlled not gate that targets the first qubit. When clear from the context we use simply notation $|0\rangle$ to denote $|0\rangle^{\otimes w}$.

1.6 Block encoding of density operators

given an s-qubit density matrix ρ and an $(a+s)$-qubit unitary G that prepares a purification of ρ as $G\left|0\right\rangle^{\otimes a}\left|0\right\rangle^{\otimes s}=\left|\rho\right\rangle$ (such that $tr_a\left|\rho\right\rangle\left\langle\rho\right|=\rho$, where tr_a denotes the trace over the first register), then the operator

$$(G^\dagger\otimes I_s)(I_s\otimes SWAP_s)(G\otimes I_s)$$

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.

Lemma 1.6.1. ***(Block-encoding of density operators)*** *Suppose that ρ is an s-qubit density operator and G is an $(a+s)$-qubit unitary that on the $\left|0\right\rangle\left|0\right\rangle$ input state prepares a purification $\left|0\right\rangle\left|0\right\rangle\rightarrow\left|\rho\right\rangle$, such that $Tr_a\left|\rho\right\rangle\left\langle\rho\right|=\rho$. Then $(G^\dagger\otimes I_s)(I_a\otimes SWAP_s)(G\otimes I_s)$ is a $(1,a+s,0)$-block encoding of ρ.*

Proof. Let r be the Schmidt-rank of ρ, let $\{\left|\psi_k\right\rangle : k\in[2^s]\}$ be an orthonormal basis, let $\{\left|\phi_k\right\rangle : k\in[r]\}$ be an orthonormal system and let $p\in[0,1]^{2^s}$ be such that $\left|\rho\right\rangle=\sum_{k=1}^{r}\sqrt{p_k}\left|\phi_k\right\rangle\left|\psi_k\right\rangle$ and $p_l=0$ for all $l\in[2^s]\notin[r]$. Then for all $i,j\in[2^s]$ we have that

$$\left\langle 0\right|^{\otimes a+s}\left\langle\psi_i\right|(G^\dagger\otimes I_s)(I_a)\otimes SWAP_s)(G^\dagger\otimes I_s)\left|0\right\rangle^{\otimes a+s}\left|\psi_j\right\rangle=$$

$$\begin{aligned}
&=\left\langle\rho\right|\left\langle\psi_i\right|(I_a\otimes SWAP_s)\left|\rho\right\rangle\left|\psi_j\right\rangle\\
&=\left(\sum_{k=1}^{r}\sqrt{p_k}\left\langle\phi_k\right|\left\langle\psi_k\right|\right)\left\langle\psi_i\right|(I_a\otimes SWAP_s)\left(\sum_{l=1}^{r}\sqrt{p_l}\left|\phi_l\right\rangle\left|\psi_l\right\rangle\right)\left|\psi_j\right\rangle\\
&=\left(\sum_{k=1}^{r}\sqrt{p_k}\left\langle\phi_k\right|\left\langle\psi_k\right|\left\langle\psi_i\right|\right)\left(\sum_{l=1}^{r}\sqrt{p_l}\left|\phi_l\right\rangle\left|\psi_j\right\rangle\left|\psi_l\right\rangle\right)\\
&=\sqrt{p_jp_i}\delta_{ij}\\
&=\left\langle\psi_i\right|\rho\left|\psi_j\right\rangle
\end{aligned}$$

Gilyen recently also showed that an implementation scheme for a POVM operator can also be easily transformed to block-encoding of the POVM operators. By an implementation scheme we mean a quantum circuit u that given input ρ and a ancilla qubits, it sets a flat qubit to 0 with probability $Tr[\rho M]$. □

1.7 Block encoding of POVM operators

One can construct block-encodings of POVM operators, given access to a unitary that implements the POVM. 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

$$|Tr(\rho M) - Tr[U(|0\rangle\langle 0|^{\otimes a} \otimes \rho)U^\dagger(|0\rangle\langle 0| \otimes I_{a+s-1}))]| \leq \epsilon$$

then $(I_a \otimes U^\dagger(CNOT \otimes I_{a+s-1})(I_a \otimes U)$ is a $(1, 1+a, \epsilon)$ bock encoding of M.

Lemma 1.7.1. ***(Block-encoding of POVM operators)*** *Suppose that U is an $a+s$ qubit unitary, which implements a POVM operator M with ϵ-precision such that for all s-qubit density operator ρ.*

$$|Tr[\rho M] - Tr[U(|0\rangle\langle 0|^{\otimes a} \otimes \rho)U^\dagger(|0\rangle\langle 0|^{\otimes 1} \otimes I_{a+s-1})]| \leq \epsilon$$

Then $(I_1 \otimes U^\dagger)(CNOT \otimes I_{a+s-1})(I_a \otimes U)$ is a $(1, 1+a, \epsilon)$-block encoding of the matrix M.

Proof. First observe that by the cyclicity of trace we have that

$$\begin{aligned} Tr[U(|0\rangle\langle 0|^{\otimes a} \otimes \rho)U^\dagger(|0\rangle\langle 0| \otimes I_{a+s-1})] &= Tr[U(|0\rangle^{\otimes a} \otimes I)\rho(\langle 0|^{\otimes a} \otimes I)U^\dagger(|0\rangle\langle 0| \otimes I_{a+s-1})] \\ &= Tr[\rho(\langle 0|^{\otimes a} \otimes I)U^\dagger(|0\rangle\langle 0| \otimes I_{a+s-1})U(|0\rangle^{\otimes a} \otimes I)] \end{aligned}$$

Together, this implies that for all ρ density operator

$$|Tr[\rho(M - (\langle 0|^{\otimes a} \otimes I)U^\dagger(|0\rangle\langle 0| \otimes I_{a+s-1})U(|0\rangle^{\otimes a} \otimes I))]| \leq \epsilon$$

which is equivalent to saying that $\|M - (\langle 0|^{\otimes a} \otimes I)(U^\dagger(|0\rangle\langle 0| \otimes I_{a+s-1})(I_a \otimes U)(|0\rangle^{\otimes 1+a} \otimes I)\| \leq \epsilon$. We can conclude by observing that

$$(\langle 0|^{\otimes a} \otimes I)U^\dagger(|0\rangle\langle 0| \otimes I_{a+s-1}U(|0\rangle^{\otimes a} \otimes I) =$$

$$= (\langle 0|^{\otimes 1+a} \otimes I)(I_a \otimes U^\dagger)(CNOT \otimes I_{a+s-1})(I_1 \otimes U)(|0\rangle^{\otimes 1+a} \otimes I)$$

□

Now we turn to a more traditional way of constructing block-encoding via state preparation. This is a common technique for example to implement quantum walks. Note that we introduce the notation $[n] - 1$ to denote the set $\{0, 1, \ldots, n-1\}$.

1.8 Block-encoding of Gram matrices

One can 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 entires are $A_{ij} = \langle\psi_i|\phi_j\rangle$, where

$$U_L |0\rangle^{\otimes a} |i\rangle = |\psi_i\rangle, \quad U_R |0\rangle^{\otimes a} |j\rangle = |\phi_j\rangle$$

Lemma 1.8.1. ***(Block encoding of gram matrices by state preparation unitaries)****. Let U_l and U_R be "state preparation" unitaries acting on $a + s$ qubits preparing the vectors $\{|\psi_i\rangle : i \in [2^s] - 1\}$, $\{|\phi_j\rangle : j \in [2^s] - 1\}$ such that*

$$U_L : |0\rangle |i\rangle \to |\psi_i\rangle$$

$$U_R : |0\rangle |j\rangle \to |\phi_j\rangle$$

Then $U = U_L^\dagger U_R$ is an $(1, a, 0)$-block encoding of the Gram matrix A such that $A_{ij} = \langle\psi_i|\phi_j\rangle$.

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 $\alpha \geq \|A\|$. We now give a few examples of block encodings of more general structured sparse matrices. See block-encoding classical data for details.

1.9 Block Encoding of s-sparse matrix

Sparse-access matrices: given a matrix $A \in \mathbb{C}^{2^w \times 2^w}$ that is s_r-row sparse and s_c-columns sparse (meaning each row/column has at most s_r or s_c nonzero entries), 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 below

$$O_r : |i\rangle |k\rangle \to |i\rangle |r_{ik}\rangle \quad \forall i \in [2^w] - 1, k \in [s_r]$$

$$O_c : |l\rangle |j\rangle \to |c_{lj}\rangle |j\rangle, \quad \forall j \in [2^w] - 1$$

$$O_A : |i\rangle |j\rangle |0\rangle^{\otimes b} \to |i\rangle |j\rangle |A_{ij}\rangle \quad \forall i, j \in [2^w] - 1$$

In the above r_{ij} is the index of the jth nonzero entry in the ith row of A (or $j + 2^w$ if there are less than i non zero entries), and c_{ij} is the index of the ith nonzero entry

in the jth 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.

Lemma 1.9.1. ***(Block-encoding of sparse-access matrices)***. *Let $A \in \mathbb{C}^{2^w \times 2^w}$ be a matrix that is s_r row sparse and s_c-column sparse, and each element of Ahas absolute value at most 1. Suppose that we have access to the following sparse-access oracles acting on two $(w+1)$ qubit registers.*

$$O_r : |i\rangle |k\rangle \to |i\rangle |r_{ik}\rangle \quad \forall i \in [2^w] - 1, k \in [s_r]$$

$$O_c : |l\rangle |j\rangle \to |c_{lj}\rangle |j\rangle \quad \forall l \in [s_c], j \in [2^w] - 1$$

where r_{ij} is the index for the jth non-zero entry of the ith row of A, or if there are less than i nonzero entries than it is $j + 2^w$, and similarly c_{ij} is the index for the ith non-zero entry of the $j$column of A, or if there are less than j non zero entries, then it is $i + 2^w$. Additionally assume that we have accesss to an oracle O_A that returns the entries of A in a binary description

$$O_A : |i\rangle |j\rangle |0\rangle^{\otimes b} \to |i\rangle |j\rangle |a_{ij}\rangle \quad \forall i, j \in [2^w] - 1$$

where a_{ij} is a b-bit binary description (for simplicity we assume here that the binary representation is exact) of the ij matrix element of A. Then we can implement a $(\sqrt{s_r s_c}, w + 3, \epsilon)$ block-encoding of A with a single use of O_r,O_c, two uses of O_A and additionally using $O(w + \log^{2.5}\left(\frac{s_r s_c}{\epsilon}\right))$ one and two qubit gates while using $\mathcal{O}(b, \log^{2.5}\left(\frac{s_r s_c}{\epsilon}\right))$ ancilla qubits.

Proof. We proceed by constructing state preparation unitaries. We will work with 3 registers the first of which is a single qubit register, and the other two registers have $(w+1)$ qubit unitary that implements the map $|0\rangle \to \sum_{k=1}^{s} \frac{|k\rangle}{\sqrt{s}}$, it is known that this operator D_s can be implemented with $\mathcal{O}(2)$ quantum gates using $\mathcal{O}(1)$ ancilla qubits. Then we define the $2(w+1)$ qubit unitary $V_L = O_r(I_{w+2} \otimes D_{s_r})SWAP_{w+1}$ such that

$$V_L : |0\rangle^{w+2} |i\rangle \to \sum_{k=1}^{s_r} \frac{|i\rangle |r_{ik}\rangle}{\sqrt{s_r}} \quad \forall i \in [2^w] - 1$$

We implement the operator $V_R = O_c(D_c \otimes I_{w+1})$ in a similar way acting as

$$V_R : |0\rangle^{w+2} |j\rangle \to \sum_{l=1}^{s_c} \frac{|c_{lj}\rangle |j\rangle}{\sqrt{s_c}} \quad \forall j \in [2^s] - 1$$

It is easy to see that the above unitaries are such that

$$\langle 0|^{w+2} \langle i| V_L^\dagger V_R |0\rangle^{w+2} |j\rangle = \frac{1}{\sqrt{s_r s_c}} \quad \text{if } a_{ij} \neq 0 \text{ and 0 otherwise}$$

Now we define $U_L = I_1 \otimes V_L$ and define U_R as performing the unitary $I_a \otimes V_R$ followed by some extra computation. After performing V_R we et a superposition of index pairs $|i\rangle |j\rangle$. Given an index pair $|i\rangle |j\rangle$ we query the matrix element $|a_{ij}\rangle$ using the oracle O_A. Then we do some elementary computations in order to implement a single qubit gate $|0\rangle \to a_{ij} |0\rangle + \sqrt{1 - |a_{ij}|^2} |1\rangle$ on the first qubit, with precision $\mathcal{O}(poly(\frac{\epsilon}{s_r s_c}))$. This can be executed with the stated complexity. Finally we also need to uncompute everything which requires one more use of O_A. This way we get a good approximation of

$$U_R : |0\rangle^{w+3} |j\rangle \to \sum_{l=1}^{s_c} \frac{(a_{c_{lj}j} |0\rangle + \sqrt{1 - |a_{c_{lj}j}} |1\rangle) |c_{lj}\rangle |j\rangle}{\sqrt{s_c}} \quad \forall j \in [2^w] - 1$$

Note that in the above method the matrix gets subnormalized by a factor of $\frac{1}{\sqrt{s_r s_c}}$. If we would know that for example $\|A\| \leq \frac{1}{2}$, then we could amplify the block-encoding in order to remove this unwanted subnormalization using singular value amplification using block encoding roughly $\sqrt{s_r s_c}$ times. However, under some circumstances one can defeat the subnormalization more efficiently by doing an amplification at the level of state preparation unitaries using a technique called "Uniform spectral gap amplification". □

Lemma 1.9.2. ***(Preamplified block-encoding of sparse-access matrices)*** *Let $A \in \mathbb{C}^{2^w \times 2^w}$ be a matrix that is s_r row sparse and s_c columns sparse, and is given using the input oracles as defined before. let a_i denote the ith row of A and similarly a_j the jth column. Let $q \in [0, 2]$ and suppose that $n_r \in [1, s_r]$ is an upper bound on $\|a_i\|^q$ and $n_c \in [1, s_c]$ is an upper bound on $\|a.j\|_{2-q}^{2-q}$.*

Let $m = max[\frac{s_r}{n_r}, s_c n_c]$. Then we can implement a $)\sqrt{\frac{1}{2n_r n_c}}, w + 6, \epsilon)$-block encoding of A with $\mathcal{O}(\sqrt{\frac{s_r}{n_r}} \log(\frac{s_r s_c}{\epsilon}))$ uses of O_r, $\mathcal{O}(\sqrt{\frac{s_c}{n_c}} \log(\frac{s_r s_c}{\epsilon}))$ uses of O_c, $\mathcal{O}(\sqrt{m} \log(\frac{s_r s_c}{\epsilon}))$ uses of O_A, and additionally using $\mathcal{O}(\sqrt{m} \log(\frac{s_r s_c}{\epsilon}) + \log^{3.5}(\frac{s_r s_c}{\epsilon}))$ one and two qubit gates while using $\mathcal{O}(b, \log^{2.5}(\frac{s_r s_c}{\epsilon}))$ ancilla qubits.

Proof. The idea is very similar to the proof of previous lemma, we implement the unitaries V_L, V_R the same way. However, we define U_R, U_L slightly differently. using

a similar method than in previous lemma we implement $\mathcal{O}(poly(\frac{\epsilon}{s_r s_c})$ approximations of the maps

$$U_L : |0\rangle^{w+4} |i\rangle \to \sum_{k=1}^{s_r} \frac{(|a_{ir_{ik}}|^{\frac{q}{2}} |0\rangle + \sqrt{1 - |a_I r_{ik}|^q} |1\rangle) |0\rangle |i\rangle |r_{ik}\rangle}{\sqrt{s_r}} \quad \forall i \in [2^w] - 1$$

$$U_R : |0\rangle^{w+4} |j\rangle \to \sum_{l=1}^{s_c} \frac{\frac{a_{c_{lj}j}}{|a_{c_{lj}j}|} |0\rangle (|a_{c_{lj}j}|^{1-\frac{q}{2}} |0\rangle + \sqrt{1 - |a_{c_{lj}j}|^{2-q}} |1\rangle) |c_{lj}\rangle |j\rangle}{\sqrt{s_c}} \quad \forall j \in [2^w] - 1$$

It is easy to see that the above unitaries are such that

$$\langle 0|^{w+4} \langle i| U_L^\dagger U_R |0\rangle^{w+4} |j\rangle = \frac{a_{ij}}{\sqrt{s_r s_c}} \quad \forall i, j \in [2^w] - 1$$

We can see that for all $i \in [2^w] - 1$ the modified row vector $\sum_{k=1}^{s_r} \frac{|a_{ir_{ik}}|^{\frac{q}{2}} |0\rangle|0\rangle|i\rangle|r_{ik}\rangle}{\sqrt{s_r}}$ has squared norm at most $\frac{n_r}{s_r}$, and a similar $\frac{n_c}{s_c}$ upper bound holds for the squared norm of the modified column vector. Also observe that

$$(|0\rangle \langle 0| \otimes I_{2w+3}) U_L (|0\rangle \langle 0|^{w+4} \otimes I_2) = \sum_{j=0}^{2^w-1} \left(\sum_{k=1}^{s_r} \frac{|a_{ir_{ik}}|^{\frac{q}{2}} |0\rangle |0\rangle |i\rangle |r_{ik}\rangle}{\sqrt{s_r}} \right) \langle 0|^{w+4} \langle j|$$

which is a singular value decomposition with the singular values being the modified row norms. Therefore we can apply singular value amplification with amplification $\gamma_r = \sqrt{\frac{s_r}{\sqrt{2} n_r}}$ and precision $\mathcal{O}(poly\left(\frac{\epsilon}{s_r s_c}\right)$)approximation of $\tilde{U}_L$ such that

$$(\langle +|\otimes|0\rangle \langle 0|\otimes I_{2w+3}) \tilde{U}_L (|+\rangle\otimes|0\rangle \langle 0|^{w+4}\otimes I_w) = \gamma_r \sum_{j=0}^{2^w-1} \left(\sum_{k=1}^{s_r} \frac{|a_{ir_{ik}}|^{\frac{q}{2}} |0\rangle |0\rangle |i\rangle |r_{ik}\rangle}{\sqrt{s_r}} \right) \langle 0|^{w+4} \langle j|$$

Similarly we apply singular value amplification with amplification $\gamma_c = \sqrt{\frac{s_c}{\sqrt{2} n_c}}$ and precision $\mathcal{O}(poly(\frac{\epsilon}{s_r s_c}))$ resulting in a $\mathcal{O}(poly(\frac{\epsilon}{s_r s_c}))$ approximation of $\tilde{U}_R$ such that

$$\langle ++| \langle 0|^{w+4} \langle i| \tilde{U}_L^\dagger \tilde{U}_R |++\rangle |0\rangle^{2+4} |j\rangle = \gamma_r \gamma_c \frac{a_{ij}}{\sqrt{s_r s_c}} = \frac{a_{ij}\sqrt{2 n_r n_c}}{} \forall i, j \in [2^w] - 1$$

Finally adding 4 Hadamard gates we can change the $|+\rangle$ states above to $|0\rangle$ states, resulting in the $\left(\frac{1}{\sqrt{2 n_r n_c}}, w + 6, \epsilon\right)$-block-encoding of A. The complexity statement follows similarly as in the previous proof, with extra observation that the singular value amplifications of U_L and U_R can be performed using degree $\mathcal{O}(\gamma_r \log(\frac{s_r s_c}{\epsilon}))$ and $\mathcal{O}(\gamma_c \log(\frac{s_r s_c}{\epsilon}))$ singular value transformations respectively. □

For $q \in [0,2]$ let us define $\mu_q(A) = \sqrt{n_q(A) n_{(2-q)}(A^T)}$, where $n_q(A) = \max_i \|a_i.\|_q^q$ is the qth power of the maximum q-norm of the rows of A. Let $A^{(q)}$ denote the matrix of the same dimensions as A, with (for complex values we define these non-integer powers using the principal value of the complex logarithm function) $A_{ij}^{(q)} = \sqrt{a_{ij}^q}$.

Lemma 1.9.3. ***(Block-encodings of matrices stored in quantum data structures)*** *(here we assume that the data structures stores the matrices with sufficient precision. Let $A \in \mathbb{C}^{2^w \times 2^w}$.*

1. *fix $q \in [0,2]$. If $A^{(q)}$ and $(A^{(2-q)})^\dagger$ are both stored in quantum accessible data structures. then there exist unitaries U_R and U_L that can be implemented in time $\mathcal{O}(poly(w \log(1/\epsilon)))$ such that $U_R^\dagger U_L$ is a $(\mu_q(A), w+2, \epsilon)$ block-encoding of A.*

2. *On the other hand, if A is stored in a quantum accessible data structure, then these exist unitaries U_R and U_L that can be implemented in time $\mathcal{O}(poly(w \log(e/\epsilon)))$ such that $U_R^\dagger U_L$ is an $(\|A\|_F, w+2, \epsilon)$-block -encoding of A.*

If, in addition to being sparse, the matrix also enjoys some additional structure, e.g. there are only a few distinct values that the matrix elements take, the complexity can be further improved. Finally, note that the sparsity dependence can be essentially quadratically improved to $(max(s_r, s_c))^{\frac{1}{2}+o(1)}$ using advanced Hamiltonian simulation techniques combined with taking the logarithm of unitaries, however the resulting subroutine may be impractical and comes with a worse precision dependence.

Example 1.9.1. Recall that real symmetric 2×2 matrix example. If we define $\phi_1 = \cos^{-1}(\alpha_1) + \cos^{-1}(\alpha_2)$ and $\phi_2 = \cos^{-1}(\alpha_1) - \cos^{-1}(\alpha_2)$, then we can verify that the block encoding for the 2×2 matrix can be factored as a product of simple unitaries i.e.

$$U_A = U_6 U_5 U_4 U_3 U_2 U_1 U_0$$

where $U_0 = U_6 = I_2 \otimes H \otimes I_2$, $U_1 = R_1 \otimes I_2 \otimes I_2, U_2 = U_4 = (I_2 \otimes E_0 + X \otimes E_1) \otimes I_2, U_3 = R_2 \otimes I_2 \otimes I_2$, and $U_5 = I_2 \otimes (E_0 \otimes I_2 + E_1 \otimes X)$. Here, H, X are the Hadamard and Pauli-X gates, $R_1 = R_y(\phi_1)$, $R_2 = R_y(\phi_2)$ are the rotation matrices along y axis. and E_0, E_1 are the projectors. The quantum circuit associated with the factorization is given in figure 1.3. Note that in this case, the quantum circuit requires 2 ancilla qubits in addition to the $n = 1$ system qubit required to match the dimension of A, which is $N = 2^n$. As a result the unitary that block encodes the 2×2 matrix A is of dimension 2^3.

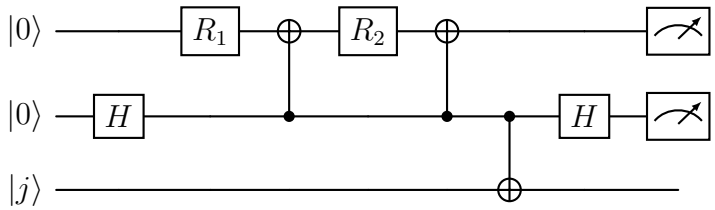


Figure 1.3: A quantum circuit for the block encoding of a 2×2 symmetric matrix A

It is possible to develop a general scheme to construct a block encoding and the corresponding quantum circuit for well structured matrices. In particular, when A is sparse with a structured sparsity pattern and the nonzero matrix elements can also be well characterized, an efficient block encoding circuit for A can be constructed.

We start from a 1-sparse matrix, i.e., there is only one nonzero entry in each row or column of the matrix. This means that for each $j \in [N]$, there is a unique $c(j) \in [N]$ such that $A_{c(j),j} \neq 0$, and the mapping c is a permutation. Then there exists a unitary O_c such that

$$O_c |j\rangle = |c(j)\rangle$$

The implementation of O_c may require the usage of some work registers that are omitted here. We also have

$$O_c^\dagger |c(j)\rangle = |j\rangle$$

We assume the matrix entry $A_{c(j),j}$ can be queried via

$$O_A |0\rangle |j\rangle = \left(A_{c(j),j} |0\rangle + \sqrt{1 - |A_{c(j),j}|^2} |1\rangle\right) |j\rangle$$

Now we construct $U_A = (I \otimes O_c)O_A$, (tensor product of two unitary matrices is unitary) and compute

$$\langle i| \langle 0| U_A |0\rangle |j\rangle = \langle 0| \langle i| \left(A_{c(j),j} |0\rangle + \sqrt{1 - |A_{c(j),j}|^2} |1\rangle\right) |c(j)\rangle = A_{c(j),j}\delta_{i,c(j)}$$

This proves that $U_A \in BE_{1,1}(A)$.

Theorem 1.9.4. *Let $c(j,l)$ be a function that gives the row index of the lth (among a list of s) non-zero matrix elements in the jth column of an s-sparse matrix $A \in \mathbb{C}^{NtimesN}$ with $N = 2^n$, where $s = 2^m$. If there exists a unitary O_x such that*

$$O_c |l\rangle |j\rangle = |l\rangle |c(j,l)\rangle$$

and a unitary O_A such that

$$O_A |0\rangle |l\rangle |j\rangle = (A_{c(j,l),j} |0\rangle + \sqrt{1 - |A_{c(j,l),j}|^2} |1\rangle) |l\rangle |j\rangle$$

then

$$U_a = (I_2 \otimes D_s \otimes I_N)(I_s \otimes O_c)(I_s \otimes D_s \otimes I_N)$$

block encodes A/s. Here D_s is called a diffusion operator and is defined as

$$D = H \otimes H \otimes \ldots \otimes H = H^{\otimes m}$$

Proof. Note that applying D_s to $|0^m\rangle$ yields

$$D_s |0^m\rangle = \frac{1}{\sqrt{s}} \sum_{l=0}^{s-1} |l\rangle$$

where $\{|l\rangle\}$ forms the computational basis in the Hilbert space defined by the m qubits. Our goal is to show that $\langle 0| \langle 0^m| \langle i| U_A |0\rangle |0^m\rangle |j\rangle = A_{ij}/s$. In order to compute the inner product $\langle 0| \langle 0^m| \langle i| U_a |0\rangle |0^m\rangle |j\rangle$, we apply D_s, O_A, O_c to $|0\rangle |0^m\rangle |j\rangle$ successively as illustrated below

$$\begin{aligned}
|0\rangle |0^m\rangle |j\rangle &\xrightarrow{I_2 \otimes D_s \otimes I_N} \frac{1}{\sqrt{s}} \sum_{l \in [s]} |0\rangle |l\rangle |j\rangle \\
&\xrightarrow{O_A} \frac{1}{\sqrt{s}} \sum_{l \in [s]} (A_{c(j,l),j} |0\rangle + \sqrt{1 - |A_{c(j,l)}|^2} |1\rangle) |l\rangle |j\rangle \\
&\xrightarrow{I_2 \otimes O_c} \frac{1}{\sqrt{s}} \sum_{l \in [s]} (A_{c(j,l),j} |0\rangle + \sqrt{1 - |A_{c(j,l),j}|^2} |1\rangle) |l\rangle |c(j,l)\rangle
\end{aligned}$$

where $[s]$ denotes the set of integers $\{0, 1, \ldots, s-1\}$. Instead of multiplying the leftmost factor $I_2 \otimes D_s \otimes I_N$ (note that D_s is Hermitian) to the last line we apply if to $|0\rangle |0^m\rangle |i\rangle$ to obtain

$$|0\rangle |0^m\rangle |i\rangle \xrightarrow{I_2 \otimes D_s \otimes I_N} \frac{1}{\sqrt{s}} \sum_{l' \in [s]} |0\rangle |l'\rangle |i\rangle$$

Finally, taking the inner product between the two results in

$$\langle 0| \langle 0^m| \langle i| U_a |0\rangle |0^m\rangle |j\rangle = \frac{1}{s} \sum_l A_{c(j,l),j} \delta_{i,c(j,l)} = \frac{1}{s} A_{ij}$$

□

The quantum circuit associated with the block encoding is as shown in figure 1.7. Note that the implementation of this block encoding requires $m+1$ ancilla qubits in addition to n system qubits. In order to turn this into an efficient quantum circuit, we need to further decompose the O_c and O_A unitaries into a sequence of quantum gates, which may not be straightforward. In the circuit shown in figure 1.3, O_A is decomposed as $O_A = U_4U_3U_2U_1$ and O_C is simply U_5. We will now show how these decompositions are constructed systematically.

Although it may be possible to construct a set of controlled quantum gates to achieve O_c and O_A oracles for a specific pair of j and l. This approach will not yield an efficient quantum circuit because the total number of quantum gates in the circuit will be of the order of $\mathcal{O}(N)$, which is exponential with respect to n. Our goal is to construct a circuit that has a gate complexity of $poly(n)$ i.e. a polynomial in n, at least for certain sparse and/or structured matrices with well defined sparsity patterns.

Note that O_c is unitary, and thus we must have $O_C^\dagger |l\rangle |c(j,l)\rangle = |l\rangle |j\rangle$. This means that for each index $i = c(j,l)$ we can recover the column index j given the value of l. This assumption is ofcourse somewhat restrictive, but it covers important cases such as banded matrices. We remark that it can generally be more difficult to explicitly construct quantum circuits for O_C in these more general query models.

Example 1.9.2. Real Symmetric 2×2 matrix We revisit the matrix considered previously once more. and view it is an s-sparse matrix with $s = 2$ even though it is dense. Therefore we can still use the recipe given in the theorem to construct a block encoding of $A/2$. We will show how explicit circuits for O_c and O_A can be constructed.

1. **The O_C circuit:** Since the second column is a down or upshift of the first one, the function $c(j,l)$m which defines the 0-based row index of the lth nonzero element in the jth column can be defined by

$$c(j,l) = j + l \bmod 2$$

for $j, l = \{0,1\}$. After enumerating all possible combinations of (l,j) input pairs and their corresponding $(l, c(j,l))$ output pairs as shown in the table 1.1, we see that O_c can simply be implemented as a CNOT gate as shown in the figure 1.4.

2. **The O_A circuit:** The basic strategy to construct a circuit for O_A defined is to use the controlled rotations to place numerical values at proper locations in U_A. Because the matrix A has at most two unique matrix elements. We need

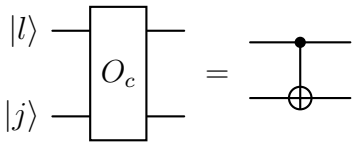


Figure 1.4: O_c circuit

l	j	l	$c(j,l)$
0	0	0	0
1	0	1	1
0	1	0	1
1	0	1	0

Table 1.1: All possible output pairs $(l, c(j,l))$ for all possible combinations of $(l, j$

two controlled rotations. In general, we need to condition the application of these rotations on the values of j and l. However, since the values of α_1 and α_2 depend solely on l, control is only required on the qubit that takes $|l\rangle$ as the input. We apply the rotation $R_u(\theta_1)$ with $\theta_1 = 2\cos^{-1}(\alpha_1)$ when $l = 0$, and $R_y(\theta_2)$ with $\theta_2 = \cos^{-1}(\alpha_2)$ when $l = 1$. These controlled rotations are combined to yield the following O_A circuit in figure 1.5. Note that the last

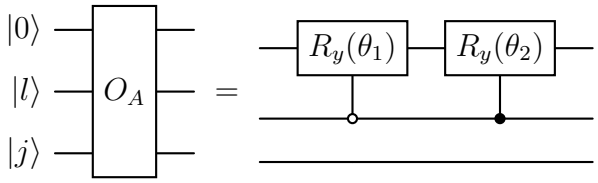


Figure 1.5: Implementation of O_A circuit

qubit is not used in this O_A circuit.

ON some quantum hardware it is preferable to use single qubit rotation and CNOT gates in place of controlled rotation gates. These are sometimes referred to as uniformly controlled rotations. In figure 1.6,we show how the controlled rotations used in Q_A circuit can be replaced by an equivalent set of uniformly controlled gates. To see that right hand side is identical to the left hand, we observe first that, if the second qubit is in the $|0\rangle$ state, the circuit on the left applies $R_y(\theta_1)$ to the first qubit. In the circuit on the right, $R_y(\phi_2)R_y(\phi_1)$ is applied to the first qubit in this case. These two operations are identical

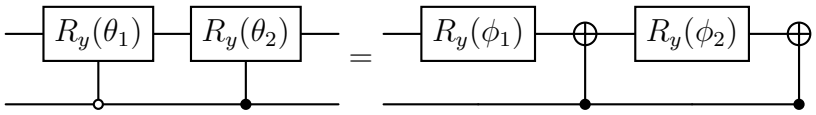


Figure 1.6: O_A circuit for uniformly controlled rotations

when $\theta_1 = \phi_1 + \phi_2$. Secondly, if the second qubit is in the $|1\rangle$ state, the left circuit applies $R_y(\theta_2)$ to the first qubit, while the right circuit applied the gate sequence $XR_Y(\phi_2)XR_Y(\phi_2)$. It follows from the identity $XR_y(\theta)X = R_Y(-\theta)$ that these operations are equivalent if $\theta_2 = \phi_1 - \phi_2$.

3. The complete circuit: Finally, we substitute circuits, together with $D_s = H$ into the figure 1.7 and obtain the complete block encoding quantum circuit for the matrix. Note that this circuit is identical to the one we have already introduced in figure 1.3.

1.9.1 Banded Circulant Matrix

For a more general s-sparse matrix, WLOG we assume each row and column ha exactly s nonzero entries (otherwise we can always treat some zero entries as nonzeros). For each column j, the row index for the lth nonzero entry is denoted by $c(j,l) = c_{j,l}$. For simplicity, we assume hat there exists a unitary O_c such that

$$O_c |l\rangle |j\rangle = |l\rangle |c(j,l)\rangle$$

Here we assume $s = 2^p$ and the first register is an p-qubit register. A necessary condition for this model is that O_C is reversible, i.e., we can have $O_c^\dagger |l\rangle |c(j,l)\rangle = |l\rangle |j\rangle$. This means that for each row index $i = c(j,l)$, we can recover the column index j given the value of l. This can be satisfied e.g.

$$c(j,l) = j + l - l_0 \quad (modN)$$

where l_0 is a fixed number. This corresponds to a banded matrix. This assumption is ofcourse somewhat restrictive. We shall discuss more general query models.

Corresponding to the equation, the matrix entries can be queried via

$$O_A |0\rangle |l\rangle |j\rangle = \left(A_{c(j,l),j} |0\rangle + \sqrt{1 - |A_{c(j,l),j}|^2} |1\rangle \right) |l\rangle |j\rangle$$

In order to construct a unitary that encodes all row indices at the same time ,we define $D = H^{\otimes p}$ (sometimes called a diffusion operator, which is a term originated

from Grover's search) satisfying

$$D\left|0^p\right\rangle = \frac{1}{\sqrt{p}}\sum_{l\in[p]}\left|l\right\rangle$$

Consider U_A given by the circuit in figure 1.7. The measurement means that to obtain $A\left|b\right\rangle$, the ancilla register should return the value 0.

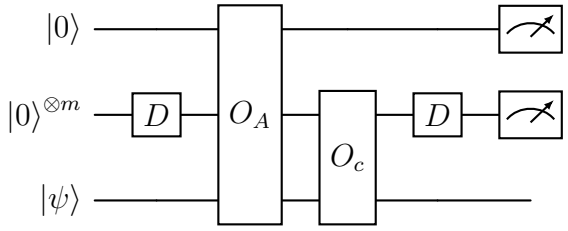


Figure 1.7: Quantum circuit for block encoding an s-sparse matrix

Proposition 1.9.5. *The circuit given in figure 1.7 defines $U_A \in BE_{s,s+1}(A)$.*

Proof. We call $\left|0\right\rangle\left|0^p\right\rangle\left|j\right\rangle$ the source state, and $\left|0\right\rangle\left|0^p\right\rangle\left|i\right\rangle$ the target state. In order to compute the inner product $\left\langle 0\right|\left\langle 0^s\right|\left\langle i\right|U_A\left|0\right\rangle\left|0^p\right\rangle\left|i\right\rangle$, we apply D, O_A, O_C to the source state accordingly as

$$\begin{aligned}
\left|0\right\rangle\left|0^p\right\rangle\left|j\right\rangle &\xrightarrow{D} \frac{1}{\sqrt{s}}\sum_{l\in[s]}\left|0\right\rangle\left|l\right\rangle\left|j\right\rangle \\
&\xrightarrow{O_A} \frac{1}{\sqrt{s}}\sum_{l\in[s]}\left(A_{c(j,l),j}\left|0\right\rangle + \sqrt{1-\left|A_{c(j,l),j}\right|^2}\left|1\right\rangle\right)\left|l\right\rangle\left|j\right\rangle \\
&\xrightarrow{O_C} \frac{1}{\sqrt{s}\sum_{l\in[s]}}\left(A_{c(j,l),j}\left|0\right\rangle + \sqrt{1-\left|A_{c(j,l),j}\right|^2}\left|1\right\rangle\right)\left|l\right\rangle\left|c(j,l)\right\rangle
\end{aligned}$$

Since we are only interested in the final state when all ancilla qubits are the 0 state, we may apply D to the target state $\left|0\right\rangle\left|0^p\right\rangle\left|i\right\rangle$ as (note that D is Hermitian)

$$\left|0\right\rangle\left|0^p\right\rangle\left|i\right\rangle \xrightarrow{D} \frac{1}{\sqrt{s}}\sum_{l'\in[s]}\left|0\right\rangle\left|l'\right\rangle\left|i\right\rangle$$

Hence the inner product

$$\left\langle 0\right|\left\langle 0^p\right|\left\langle i\right|U_A\left|0\right\rangle\left|0^p\right\rangle\left|j\right\rangle = \frac{1}{s}\sum_{l}A_{c(j,l),j}\delta_{i,c(j,l)} = \frac{1}{s}A_{ij}$$

□

1.10 Hermitian Block Encoding

So far we have considered general s-sparse matrices. Note that if A is Hermitian matrix, its (α, m, ϵ)-block-encoding U_A does not need to be Hermitian. Even if $\epsilon = 0$, we only have that the upper-left n-qubit block of U_A is Hermitian. For instance, even the block encoding of a Hermitian diagonal matrix may not be Hermitian. On the other hand,there are cases when $U_A^\dagger = U_A$ is indeed a Hermitian matrix, and hence the definition:

Definition 1.10.1. (Hermitian Block Encoding). Let U_A be an (α, m, ϵ)-block-encoding of A. If U_A is also Hermitian, then it is called an (α, m, ϵ)-Hermitian-block-encoding of A. When $\epsilon = 0$, it is called an (α, m)-Hermitian-block-encoding. The set of all (α, m, ϵ)-Hermitian-block-encoding of A is denoted by $HBE_{\alpha,m}(A, \epsilon)$, and we define $HBE_{\alpha,m}(A) = HBE(A, 0)$.

The Hermitian block encoding provides the simplest scenario of the qubitization process.

1.11 Query models for general sparse matrices

If we query the oracle, the assumption that for each l the value of $c(j, l)$ is unique for all j seems unnatural for constructing general sparse matrices. So we consider an alternative method for construct the block encoding of a general sparse matrix as below.

Again without loss of generality we assume that each row/column has at most $p = 2^s$ non zero entires, and that we have access to the following two (2n)-qubit oracles.

$$O_r |l\rangle |i\rangle = |r(i, l)\rangle |i\rangle$$
$$O_c |l\rangle |j\rangle = |c(j, l)\rangle |j\rangle$$

Here $r(i, l)$,$c(j, l)$ gives the l-th nonzero entry in the i-th row and j-th column, respectively. It should be noted that although the index $l \in [p]$, we should expand it into an n-qubit state (e.g. let l take the last s qubits of the n-qubit register following the binary representation of integers). The reason for such an expansion, and that we need two oraclesO_c, O_r will be seen shortly.

Similar to the discussion before, we need a diffusion operator satisfying

$$D |0^n\rangle = \frac{1}{\sqrt{s}} \sum_{l \in [p]} |l\rangle$$

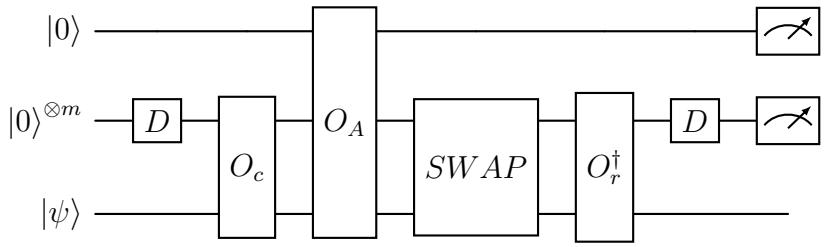


Figure 1.8: Quantum circuit for block encoding of general sparse matrices

This can be implemented using Hadamard gates as

$$D = I_{n-s} \otimes H^{\otimes s}$$

We assume that the matrix entries are queried using the following oracle using controlled rotations

$$O_A |i\rangle |j\rangle = \left(A_{ij} |0\rangle + \sqrt{1 - |A_{ij}|^2} |1\rangle \right) |i\rangle |j\rangle$$

where the rotation is controlled by both row and column indices. However, if $A_{ij} = 0$ for some i, j, the rotation can be arbitrary, as there will be no contribution due to the usage of O_r, O_c.

Proposition 1.11.1. *The circuit shown in figure 1.8 defines $U_A \in BE_{s,n+1}(A)$.*

Proof. We apply the first four gate sets to the source state

$$\begin{aligned} |0\rangle |0^n\rangle |j\rangle &\xrightarrow{D} \xrightarrow{O_c} \xrightarrow{O_A} \frac{1}{\sqrt{p}} \sum_{l \in [p]} \left(A_{c(j,l),j} |0\rangle + \sqrt{1 - |A_{c(j,l),j}|^2} |1\rangle \right) |c(j,l)\rangle |j\rangle \\ &\xrightarrow{SWAP} \frac{1}{\sqrt{p}} \sum_{l \in [p]} \left(A_{c(j,l),j} |0\rangle + \sqrt{1 - |A_{c(j,l),j}|^2} |1\rangle \right) |j\rangle |c(j,l)\rangle \end{aligned}$$

we then apply D and O_r to the target state

$$|0\rangle |0^n\rangle |i\rangle \xrightarrow{D} \xrightarrow{O_r} \frac{1}{\sqrt{p}} \sum_{l' \in [p]} |0\rangle |r(i,l')\rangle |i\rangle$$

Then the inner product gives

$$\begin{aligned} \langle 0| \langle 0^n| \langle i|U_A|0\rangle |0^n\rangle |j\rangle &= \frac{1}{p} \sum_{l,l'} A_{c(j,l),j} \delta_{i,c(j,l)} \delta_r(j,l'), j \\ &= \frac{1}{p} \sum_{l} A_{c(j,l),j} \delta_{i,c(j,l)} = \frac{1}{s} A_{ij} \end{aligned}$$

Here we have used that there exists a unique l such that $i = c(j, l)$, and a unique l' such that $j = r(i, l')$. □

We remark that the quantum circuit in figure 1.8 is essentially the construct, which gives a $(s, n + 3)$-block-encoding. The construct above slightly simplifies the procedure and saves two extra qubits (used to marke whether $l \geq s$).

Next we consider the Hermitian block encoding of a s-sparse Hermitian matrix. Since A is Hermitian, we only need one oracle to query the location of th nonzero entries.

$$O_c |l\rangle |j\rangle = |c(j, l)\rangle |j\rangle$$

Here $c(j, l)$ gives the l-th nonzero entry in the j-th column. It can also be interpreted as the l-th nonzero entry in the ith column. Again the first register needs to be interpreted as an n-qubit register. The diffusion operator is the same as in the equation.

Unlike all discussions before, we introduce two signal qubits and a quantum state in the computational basis take the form $|a\rangle |i\rangle |b\rangle |j\rangle$, where $a, b \in \{0, 1\, i, j \in [N]$. In other words, we may view $|a\rangle |i\rangle$ as the first register, and $|b\rangle |j\rangle$ as the second register. The $(n + 1)$-qubit SWAP gate is defined as

$$SWAP |a\rangle |i\rangle |b\rangle |j\rangle = |b\rangle |j\rangle |a\rangle |i\rangle$$

To query matrix entries, we need access to the square root of A_{ij} as (note that act on the second single-qubit register)

$$O_A |i\rangle |0\rangle |j\rangle = |i\rangle \left(A_{ij} |0\rangle + \sqrt{1 - |A_{ij}|} |1\rangle \right) |j\rangle$$

The square root operation is well defined if $A_{ij} \geq 0$ for all entries. If A has negative (or complex) entries, we first write $A_{ij} = |A_{ij} e^{\iota\theta_{ij}}, \theta_{ij} \in [0, 2\pi)$, and the square root is uniquely defined as $\sqrt{A_{ij}} = \sqrt{|A_{ij}|} e^{\iota\theta_{ij}/2}$.

Proposition 1.11.2. *Figure 1.9 defines $U_A \in HBE_{s,n+2}(A)$.*

Proof. Apply the first four gate sets to the source state gives

$$\begin{aligned} |0\rangle |0^n\rangle |0\rangle |j\rangle \xrightarrow{D} \xrightarrow{O_c} \xrightarrow{O_A} & \frac{1}{\sqrt{s}} \sum_{l \in [s]} |0\rangle |c(j, l)\rangle \left(\sqrt{A_{c(j,l),j}} |0\rangle \, sqrt1 - |A_{c(j,l),j} |1\rangle \right) |j\rangle \\ \xrightarrow{SWAP} & \frac{1}{\sqrt{s}} \sum_{l \in [s]} \left(\sqrt{A_{c(i,l'),i}} |0\rangle + \sqrt{1 - |A_{c(i,l'),i}|} |1\rangle \right) |i\rangle \end{aligned}$$

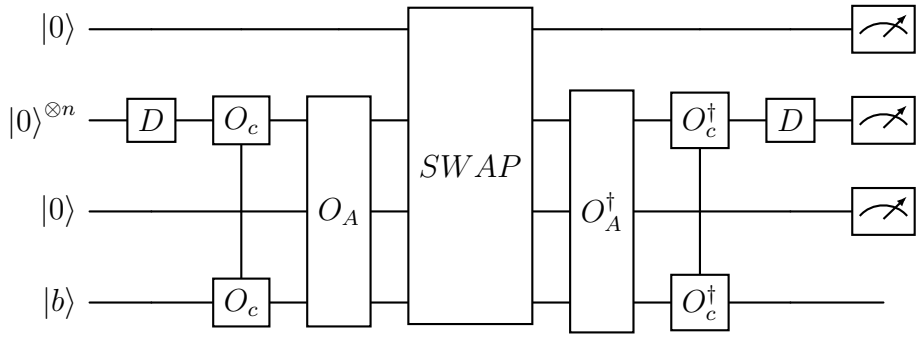


Figure 1.9: Quantum circuit for Hermitian Block encoding of a general Hermitian matrix

Apply the last three gate sets to the target state

$$|0\rangle |0^n\rangle |0\rangle |i\rangle \xrightarrow{D}\xrightarrow{O_c}\xrightarrow{O_A} \frac{1}{\sqrt{s}} \sum_{l\in[s]} |0\rangle |c(i,l')\rangle \left(\sqrt{A_{c(i,l'),i}}\,|0\rangle + \sqrt{1-|A_{c(i,l'}|}\,|1\rangle\right) |i\rangle$$

Finally, take the inner product as

$$\begin{aligned}\langle 0| \langle 0^n| \langle 0| \langle i|\, U_A\, |0\rangle |0^n\rangle |0\rangle |j\rangle & \\ &= \frac{1}{s} \sum_{l,l'} \sqrt{A_{c(j,l'),j}} \sqrt{A^*_{c(i,l'),i}} \delta_{i,c(j,l)} \delta_{c(i,l'),j} \\ &= \frac{1}{s} \sqrt{A_{ij} A^*_{ji}} \sum_{l,l'} \delta_{i,c(j,l)} \delta_{c(i,l'),j} = \frac{1}{s} A_{ij}\end{aligned}$$

In this equality, we have used that A is Hermitian: $A_{ij} = A^*_{ji}$, and there exists a unique l such that $i = c(j,l)$, as well as a unique l' such that $j = c(i,l')$.

The quantum circuit in figure 1.9 is essentially construction in. The relation will quantum walks will be discussed further. □

1.12 Examples

The cost of data input can dominate the run0time of quantum algorithms. Here, we consider data input of arithmaetically structured matrices via block encoding circuits, the input model for the quantum singular value transform and related algorithms. We demonstrate how to construct block encoding circuits based on arithmetic description

of the sparsity and pattern of repeated values of a matrix. We present schemes yielding different subnormalizations of the block encoding; a comparison shows that the best choice depends on the specific matrix. The resulting circuit reduce flag qubit number according to sparsity, and data loading cost according to repeated values, leading to an exponential improvement for certain matrices. We dive examples of our block encoding schemes to a few families of matrices, including Toeplitz and tridiagonal matrices.

Currently, the number of quantum algorithms and application use cases known to provide an exponential advantage are rather limited. The incpetion of the quantum singular value transformation (QSVT) has led to a new perspecitve on quantum algorithms. In what has been terms the "grand unification of quantum algorithms", many previously known quantum algorithms have been reformulated within the framework of QSVT, including matrix inversion, phase estimation, Hamiltonian simulation, and amplitude amplification. The core of the QSVT is a polynomial transformation of an input matrix's singular values. Different choices of polynomials and input matrices give rise to these various applications. In order to run a quantum algorithm in a real-world setting, data input (into the quantum computer) and data output (read-out) are important steps and can seerly limit any speed up provided by the quantum algorithm itself. In this, we will study how to input structed data effficiently and provide a scheme that facilitates the construction of explicit quantum circuits for input structured data matrices.

In QSVT based algorithms, the input model for matrices of data is of block encoding,, where an input matrix of data A could be non-unitary and there is no quantum circuit that could implement the operator A directly. The existence of an embedding is not sufficient to input the data matrix A. Rather, U must be implemented as a quantum cirucits and expressed in an elementary quantum gate set in order to run it on a quiantum computer. Finidng such a quantum circuit will typically further increase the subnormalization α and the dimension of the junk blocks. In a quantum circuit, the block encoding can be written as follows:

1.13 Resource cost -gates and 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 resources requirements depend entirely on their circuit level implementation without any additional overhead for begin 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, as those two subroutines typically dominate the resource requirements. For example, the best-known circuit that implement block-encoding matrices of classical data for general, dense $N \times N$ matrices uses $\mathcal{O}(N \log(1/\epsilon))$ qubits to achieve minimum T-gate count (which also scales as $\mathcal{O}(N \log(1/\epsilon))$, or a larger $\mathcal{O}(N^2)$ $\mathcal{O}(N^2)$ number of qubits to achieve minimum T-gate depth (which scales as $\mathcal{O}(\log N + \log(1/\epsilon))$. In the sparse-access model, one can use $\mathcal{O}(w + \log^{2.5}(s_r s_c/\epsilon))$ one and two qubit gates, and $\mathcal{O}(b + \log^{2.5}(s_r s_c/\epsilon))$ ancilla qubits, 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 QRAM primitive. Assuming appropriate binary representations of the numbers A_{ij}, the exponents of the above logarithms can be reduced to 1 using some techniques.

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.

Important Note

A block-encoded matrix A must have norm $\|A\| \leq 1$, or otherwise the matrix must first be normalized, and one must take care to keep track of normalization throughout the computation. In the definition of block-encodings shown above, the parameter α plays the role of normalizing A. Note that often above constructions are suboptimal in the sense that $\alpha >> \|A\|$, which can lead to increase complexity.
For a given desired block encoding, there can be several independent, yet equally valid implementations, and one can sometimes optimize for various resources when building the block-encoding. For example, many block-encoding strategies require a set 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. One can also consider a more general definition as "projected unitary encodings" which allows using an arbitrary subspace, rather than just a state-indexed block.

1.14 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 no-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 (psuedo) inversion that can be used for solving large linear system of equations or more generally least-squares regression problems.

- 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.

Chapter 2

Linear Combination of Unitaries

2.1 Introduction

In practical applications, we are often not interested in constructing the Chebyshev Polynomial of A, but linear combination of Chebyshev polynomials. For instance, the matrix inversion problem can be solved by expanding $f(x) = x^{-1}$ using a linear combination of Chebyshev polynomials on the interval $[-1, -\kappa^{-1}] \cup [\kappa^{-1}, 1]$. Here we assume $\|A\| = 1$ and $\|A\|\|A^{-1}\|$ is the condition number. One way to implement this is via a quantum primitive called the linear combination of unitaries.

Let $T = \sum_{i=0}^{K-1} \alpha_i U_i$ be the linear combination of unitary matrices U_i. For simplicity let $K = 2^a$, and $\alpha_i > 0$ (Without loss of generality we can absorb the phase of α_i into the unitary U_i). Then

$$U = \sum_{i \in [K]} |i\rangle \langle i| \otimes U_i$$

Let V be a unitary operator satisfying

$$V |0^a\rangle = \frac{1}{\sqrt{\|\alpha\|_1}} \sum_{i \in [K]} \sqrt{\alpha_i} |i\rangle$$

and V is called the prepare oracle. The 1-norm of the coefficients is given by

$$\|\alpha\|_1 = \sum_i |\alpha_i|$$

In the matrix form

$$V = \frac{1}{\sqrt{\|\alpha\|_1}} \begin{bmatrix} \sqrt{\alpha_0} & \dots & \dots & \dots \\ \vdots & \dots & \ddots & \dots \\ \sqrt{\alpha_{K-1}} & \dots & \dots & \dots \end{bmatrix}$$

where the first basis is $|0^m\rangle$, and all other basis functions are orthogonal to it. Then

$$V^\dagger = \frac{1}{\sqrt{\|\alpha\|_1}} \begin{bmatrix} \sqrt{\alpha_0} & \dots & \sqrt{\alpha_{K-1}} \\ \dots & \dots & \dots \\ \vdots & \ddots & \dots \\ \dots & \dots & \dots \end{bmatrix}$$

Then T can be implemented using the unitary given as follows (called the LCU lemma).

Lemma 2.1.1. ***(LCU)*** *Define $W = (V^\dagger \otimes I_n)U(V \otimes I_n)$, then for any $|\psi\rangle$,*

$$W |0^a\rangle |\psi\rangle = \frac{1}{\|\alpha\|_1} |0^a\rangle T |\psi\rangle + |\tilde{\perp}\rangle$$

where $|\tilde{\perp}\rangle$ is an unnormalized state satisfying

$$(|0^a\rangle \langle 0^a| \otimes I_n) |\tilde{\perp}\rangle = 0$$

In other words, $W \in BE_{\|\alpha\|_1,a}(T)$

Proof.

$$
\begin{aligned}
W\left|0^a\right\rangle\left|\psi\right\rangle &= (V^\dagger \otimes I_n)U(V\otimes I_n)\left|0^a\right\rangle\left|\psi\right\rangle \\
(V^\dagger \otimes I_n)U(V\otimes I_n)\left|0^a\right\rangle\left|\psi\right\rangle &= (V^\dagger \otimes I_n)U\frac{1}{\sqrt{\|\alpha\|_1}}\sum_i \sqrt{\alpha_i}\left|i\right\rangle\left|\psi\right\rangle \\
&= (V^\dagger \otimes I_n)\left(\sum_j \left|j\right\rangle\left\langle j\right|\otimes U_j\right)\frac{1}{\sqrt{\|\alpha\|_1}}\sum_i \sqrt{\alpha_i}\left|i\right\rangle\left|\psi\right\rangle \\
&= (V^\dagger \otimes I_n)\left(\sum_j\sum_i \frac{\sqrt{\alpha_i}}{\sqrt{\|\alpha\|_1}}\left|j\right\rangle\left\langle j|i\right\rangle\otimes U_j\left|\psi\right\rangle\right) \\
&= (V^\dagger \otimes I_n)\left(\frac{1}{\sqrt{\|\alpha\|_1}}\sum_i \sqrt{\alpha_i}\left|i\right\rangle\otimes U_i\left|\psi\right\rangle\right) \\
&= \frac{1}{\sqrt{\|\alpha\|_1}}(V^\dagger \otimes I_n)\left(\sum_i \sqrt{\alpha_i}\left|i\right\rangle\otimes U_i\left|\psi\right\rangle\right) \\
&= \frac{1}{\sqrt{\|\alpha\|_1}}\left(\sum_i \sqrt{\alpha_i}V^\dagger\left|i\right\rangle\otimes U_i\left|\psi\right\rangle\right) \\
&= \frac{1}{\|\alpha\|_1}\left(\sum_i \sqrt{\alpha_i}\begin{bmatrix}\sqrt{\alpha_0} & \cdots & \sqrt{\alpha_{K-1}}\\ \cdots & \cdots & \cdots \\ \vdots & \ddots & \cdots \\ \cdots & \cdots & \cdots\end{bmatrix}\left|i\right\rangle\otimes U_i\left|\psi\right\rangle\right) \\
&= \frac{1}{\|\alpha\|_1}\left(\sqrt{\alpha_0}\begin{bmatrix}\sqrt{\alpha_0}\\ \cdots \\ \vdots \\ \cdots\end{bmatrix}\otimes U_0\left|\psi\right\rangle + \ldots + \sqrt{\alpha_{K-1}}\begin{bmatrix}\sqrt{\alpha_{K-1}}\\ \cdots \\ \vdots \\ \cdots\end{bmatrix}\otimes U_{K-1}\left|\psi\right\rangle\right) \\
&= \frac{1}{\|\alpha\|_1}\left(\sum_i\left(\begin{bmatrix}\alpha_i\\ 0\\ \vdots\\ 0\end{bmatrix}+\begin{bmatrix}0\\ \cdots\\ \vdots\\ \cdots\end{bmatrix}\right)\otimes U_i\left|\psi\right\rangle\right) \\
&= \frac{1}{\|\alpha\|_1}\left(\sum_i\left(\begin{bmatrix}\alpha_i\\ 0\\ \vdots\\ 0\end{bmatrix}+\left|\perp_i\right\rangle\right)\otimes U_i\left|\psi\right\rangle\right)
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{\|\alpha\|_1} \left(\sum_i \left(\alpha_i \left| 0^a \right\rangle + \left| \perp_i \right\rangle \right) \otimes U_i \left| \psi \right\rangle \right) \\
&= \frac{1}{\|\alpha\|_1} \left(\sum_i \alpha_i \left| 0^a \right\rangle \otimes U_i \left| \psi \right\rangle + \left| \perp_i \right\rangle \otimes U_i \left| \psi \right\rangle \right) \\
&= \frac{1}{\|\alpha\|_1} \left(\sum_i \alpha_i \left| 0^a \right\rangle U_i \left| \psi \right\rangle + \left| \tilde{\perp}_i \right\rangle \right) \\
&= \frac{1}{\|\alpha\|_1} \left(\sum_i \alpha_i \left| 0^a \right\rangle U_i \left| \psi \right\rangle + \sum_i \left| \tilde{\perp}_i \right\rangle \right) \\
&= \frac{1}{\|\alpha\|_1} \left(\left| 0^a \right\rangle \left(\sum_i \alpha_i U_i \right) \left| \psi \right\rangle + \left| \tilde{\perp} \right\rangle \right) \\
&= \frac{1}{\|\alpha\|_1} \left| 0^a \right\rangle T \left| \psi \right\rangle + \left| \tilde{\perp} \right\rangle
\end{aligned}
$$

Thus, upon measuring the first register of a qubits in $|0\rangle$ state will result in the second register being in the state $\frac{1}{\|\alpha\|_1} T |\psi\rangle$ state. □

The LCU lemma is a useful quantum primitive, as it states that the number of ancilla qubits needed only depends logarithmically on K, the number of terms in the linear combination. Hence it is possible to implement the linear combination of a very large number of terms efficiently. From a practical perspective, the select and prepare oracles uses multi-qubit controls, and can be difficult to implement. If implemented directly, the number of multi-qubit controls again depends linearly on K and is not desirable. Therefore an efficient implementation using LCU (in terms of the gate complexity) also requires additional structures in the prepare and select oracles.

If we apply W to $|0^a |\psi\rangle\rangle$ and measure the ancilla qubits, then the probability of obtaining the outcome 0^a in the ancilla qubits (and therefore obtaining the state $T |\psi\rangle / \|T |\psi\rangle \|$ in the system register) is $(\|T |\psi\rangle \| / \|\alpha\|_1)^2$. The expected number of repetition needed to succeed is $(\|\alpha\|_1 / \|T |\psi\rangle \|)^2$. Now we demonstrate that using the amplitude amplification (AA), this number can be reduced to $\mathcal{O}(\|\alpha\|_1 / T |\psi\rangle \|)$.

Remark. **(Alternate construction of the prepare oracle)**. In some applications it may not be convenient to absorb the phase of α_i to the select oracle. In such a case, we may modify the prepare oracle instead. If $\alpha_i = r_i e^{\iota \theta_i}$ with $r_i > 0, \theta_i \in [0, 2\pi)$, we

can define $\sqrt{\alpha_i} = \sqrt{r_i}e^{\iota\theta_i/2}$, and V is defined as earlier. However instead of $V^\dagger$, we need to introduce

$$\tilde{V} = \frac{1}{\sqrt{\|\alpha\|_1}} \begin{pmatrix} \sqrt{\alpha_0} & \cdots & \sqrt{\alpha_{K-1}} \\ \cdots & \cdots & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots \end{pmatrix}$$

Then following the same proof as earlier, we can show that $W = (\tilde{V} \otimes I_n)U(V \otimes I_n) \in BE_{\|\alpha\|_1,a}(T)$.

Remark. **(Linear combination of non-unitaries)**. Using the block encoding technique, we may immediately obtain linear combination of general matrices that are not unitaries. However, with some abuse of notation, the ter "LCU" will be used whether the terms to be combined are unitaries or not. In other words, the term "linear combination of unitaries" should be loosely interpreted as "Linear combination of things" (LCT) in many contexts.

For matrices given as linear combination of unitary operators (LCU), we can block-encode the matrix using the LCU technique. We provide a full description 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 $U = PREPARE^\dagger \cdot SELECT \cdot PREPARE$. The Hamiltonians of physical system 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 and condensed matter physics.

2.2 Manipulating block-encodings

We use a simple but powerful method for implementing linear combination of unitary operators on a quantum computer. This technique can have exponentially improving the precision of Hamiltonian simulation. Later it was adapted for exponentially improving the precision of quantum linear equation solving. Here we present this method from the perspective of block-encoded matrices.

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.

We will consider the case of two operators A and B with straightforward generalizations to additional operators. 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.

First we define state preparation unitaries in order to conveniently state our result in the following lemma.

Definition 2.2.1. (State preparation pair) Let $y \in \mathbb{C}^m$ and $\|y\|_a \leq \beta$, the pair of unitaries (P_L, P_R) is called a (β, b, ϵ)-state preparation pair if $P_L |0\rangle^{\otimes b} = \sum_{j=0}^{2^b-1} c_j |j\rangle$ and $P_R |0\rangle^{\otimes b} = \sum_{j=1}^{2^b-1} d_j |j\rangle$ such that $\sum_{j=0}^{m-1} |\beta(c_j^* d_j) - y_j| \leq \epsilon$ and for all $j \in m, \ldots, 2^b - 1$ we have $c_j^* d_j = 0$.

Now we show how to implement a block-encoding of block-encoded operators.

2.2.1 Products

In general if we want to take the product of two block encoded matrices we need to treat their ancilla qubits separately. In this case as the following lemma shows the errors simply add up and the block encoding does not introduce any additional errors.

Lemma 2.2.1. ***(Product of block-encoded matrices)*** *If U is an (α, a, δ)-block-encoding of an s qubit operator A, and V is an (β, b, ϵ)-block-encoding of an s qubit operator B then $(I_b \otimes U)(I_a \otimes V)$ is an $(\alpha\beta, a + b, \alpha\epsilon + \beta\delta)$-block-encoding of AB.*

Proof.

$$\begin{aligned}
&\|AB - \alpha\beta((\langle 0|^{\otimes a+b} \otimes I)(I_b \otimes U)(I_a \otimes V)(|0\rangle^{\otimes a+b)} \otimes I)\| \\
&= \|AB - \alpha(\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I)\beta(\langle 0|^{\otimes b} \otimes I)V(|0\rangle^{\otimes b} \otimes I)\| \\
&= \|AB - \tilde{A}B + \tilde{A}B - \tilde{A}\tilde{B}\| \\
&= \|(A - \tilde{A})B + \tilde{A}(B - \tilde{B})\| \\
&\leq \|A - \tilde{A}\|\|B\| + \|\tilde{A}\|\|B - \tilde{B}\| \\
&\leq \delta\|B\| + \alpha\epsilon
\end{aligned}$$

Using the fact that $\|B\| \leq \beta + \epsilon$, we get,

$$\begin{aligned}
\|AB - \tilde{A}\tilde{B}\| &\leq \alpha\epsilon + \delta(\beta + \epsilon) \\
&\leq \alpha\epsilon + \beta\delta + \delta\epsilon
\end{aligned}$$

Now, ignoring $\delta\epsilon$ since it will be a very small quantity, we get,

$$\|AB - \tilde{A}\tilde{B}\| \leq \alpha\epsilon + \beta\delta$$

where $\tilde{A} = \alpha(\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I)$ and $\tilde{B} = \beta(\langle 0|^{\otimes b} \otimes I)V(|0\rangle^{\otimes b} \otimes I)$. □

In the special case when the encoded matrices are unitaries and their block-encoding does not use any extra scaling factor, then we might reuse the ancilla qubits, however it introduces an extra error term, which can be bounded by geometric mean of the two input error bounds.

Lemma 2.2.2. ***(Product of block-encoded matrices)*** *If U is an $(1, a, \delta)$-block-encoding of an s-qubit operator A, and V is an $(1, a, \epsilon)$-block-encoding of an s-qubit operator B then UV is a $(1, a, \delta + \epsilon + 2\sqrt{\delta\epsilon})$-block-encoding of the unitary operator AB.*

Proof. It is enough to show that for all s-qubit pure states $|\phi\rangle, |\psi\rangle$ we have that

$$\langle\phi| AB |\psi\rangle - \langle\phi| (\langle 0|^{\otimes a} \otimes I)UV(|0\rangle^{\otimes a} \otimes I) |\psi\rangle | \leq \delta + \epsilon + 2\sqrt{\delta\epsilon}$$

Observe that

$$\begin{aligned}
&\langle\phi| (\langle 0|^{\otimes a} \otimes I)UV(|0\rangle^{\otimes a}) |\psi\rangle \\
&= \langle\phi| (\langle 0|^{\otimes a} \otimes I)U((|0\rangle \langle 0|^{\otimes a} \otimes I) + ((I - |0\rangle \langle 0|^{\otimes a}) \otimes I)) + ((I - |0\rangle \langle 0|^{\otimes a} \otimes I))V(|0\rangle^{\otimes a} \otimes I) |\psi\rangle \\
&= \langle\phi| (\langle 0|^{\otimes a})U(|0\rangle^{\otimes a} \otimes I)(\langle 0|^{\otimes a} \otimes I)V(|0\rangle^{\otimes a} \otimes I) |\psi\rangle \\
&+ \langle\phi| (\langle 0|^{\otimes a} \otimes I)U((I - |0\rangle \langle 0|^{\otimes a}) \otimes I)V(|0\rangle^{\otimes a} \otimes I) |\psi\rangle
\end{aligned}$$

Now we can see that similarly, we have

$$\begin{aligned}
&| \langle\phi| AB |\psi\rangle - \langle\phi| (\langle 0|^{\otimes a} \otimes I)(\langle 0|^{\otimes a} \otimes I)V(|0\rangle^{\otimes a} \otimes I) |\psi\rangle | \\
&= | \langle\phi| (AB - (\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I)(\langle 0|^{\otimes a} \otimes I)V(|0\rangle^{\otimes a} \otimes I)) |\psi\rangle \\
&\leq \|AB - (\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I)(\langle 0|^{\otimes a} \otimes I)V(|0\rangle^{\otimes a} \otimes I)\| \\
&\leq \delta + \epsilon
\end{aligned}$$

Finally note that

$$\begin{aligned}
&| \langle\phi| (\langle 0|^{\otimes a} \otimes I)U(I - |0\rangle \langle 0|^{\otimes a}) \otimes I)V(|0\rangle^{\otimes a} \otimes I) |\psi\rangle | \\
&= | \langle\phi| (\langle 0|^{\otimes a} \otimes I)U((I - |0\rangle \langle 0|^{\otimes a}) \otimes I)^2V(|0\rangle^{\otimes a} \otimes I) |\psi\rangle | \\
&\leq \|((I - |0\rangle \langle 0|^{\otimes a}) \otimes I)U(|0\rangle^{\otimes a} \otimes I) |\phi\rangle \| \cdot \|((I - |0\rangle \langle 0|^{\otimes a}) \otimes I)V(|0\rangle^{\otimes a} \otimes I) |\psi\rangle \| \\
&= \sqrt{1 - \|(|0\rangle \langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I) |\phi\rangle \|^2} \cdot \sqrt{1 - \| |0\rangle \langle 0|^{\otimes a} \otimes I)V(|0\rangle^{\otimes a} \otimes I)V(|0\rangle^{\otimes a} \otimes I) |\psi\rangle \|^2} \\
&\leq \sqrt{1 - (1 - \delta)^2} \cdot \sqrt{1 - (1 - \epsilon)^2} \\
&\leq 2\sqrt{\delta\epsilon}
\end{aligned}$$

□

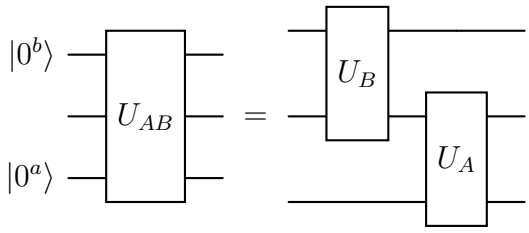


Figure 2.1: Implementing the block-encoding U_{AB} of AB that acts on s qubits. The cost is $a + b$ ancilla qubits, and 1 call to each of U_A, U_B.

The following corollary suggest that if we multiply together multiple block-encoded unitaries, the error may grow super-linearly, but it increases at most quadratically with the number of factors in the product.

Corollary 2.2.3. ***(Product of multiple block-encoded unitaries)*** *Suppose that U_j is an $(1, a, \epsilon)$ block encoding of an s qubit unitary operator W_j for all $j \in [K]$. Then $\prod_{j=1}^{K} U_j$ is an $(1, a, 4K^2\epsilon)$-block-encoding of $\prod_{j=1}^{K} W_j$.*

Proof. First observe that for the product of two matrices we get the precision bound 4ϵ by the above lemma. If $K = 2^k$ for some $k \in \mathbb{N}$. Then we can apply the above observation in a recursive fashion in a binary tree structure, to get the upper bound $4^k\epsilon$ on the precision, and observe that $4^k = K^2$.

If $2^{k-1} \leq K \leq 2^k$ we can just add identity operators so that we have 2^k matrices to multiple, which gives the precision bound $4^k\epsilon \leq 4^{1+\log_2 K}\epsilon = 4K^2\epsilon$. □

The operator $U_{AB} = (I_b \otimes U_A)(U_B \otimes I_a)$ is an $(\alpha\beta, a+b, \alpha\epsilon_b + \beta\epsilon_a)$-block-encoding of AB, where I_x denotes the identity operator on x qubits. For example, if $a = b$, this construction uses twice as many ancilla qubits for block-encoding the product compared to the block-encoding of individual matrices. In fact we can assume without loss of generality that $a = b$ (by taking the max of the two) and improve the construction using the circuit in figure 2.1.

2.2.2 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 $A \otimes B$.

2.2.3 Linear Combinations

Linear combinations of block-encodings can be viewed as a generalization of the linear combination of Unitaries (LCU). We wish to implement a block-encoding of $\sum_{i=0}^{L-1} c_i A_i$, where $c_i \in \mathbb{R}$(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 latter can be taken as $m = \max_i m_i$.

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

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

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

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

We then have the following result:

$$(\langle 0^{\lceil \log_2(L) \rceil}| \otimes I) PREPARE^\dagger \cdot SELECT \cdot PREPARE (|0^{\lceil \log_2(L) \rangle} \otimes I) = \frac{1}{\lambda} \sum_{l=0}^{L-1} c_i U_i$$

i.e. $U_{LC} = PREPARE^\dagger \cdot SELECT \cdot PREPARE$ is a $(\lambda, \lceil \log_2(L) \rceil, 0)$-block-encoding of the LCU $\sum_I c_i U_i$. This is the standard LCU trick, 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:

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

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

Lemma 2.2.4. ***(Linear combination of block-encoded matrices)*** *Let $A = \sum_{j=1}^{m} y_j A_j$ be an s qubit operator and $\epsilon \in \mathbb{R}_+$. Suppose that (P_L, P_R) is a (β, b, ϵ_1) state preparation pair for y. $W = \sum_{j=0}^{m-1} |j\rangle\langle j| \otimes U_j + ((I - \sum_{j=0}^{m-1} |j\rangle\langle j| \otimes I_a \otimes I_s)$ is an $s+a+b$ qubit unitary such that for all $j \in 0, \ldots, m$ we have that U_j is an (α, a, ϵ_2) block-encoding of A_j. Then we can implement a $(\alpha\beta, a+b, \alpha\epsilon_1+\alpha\beta\epsilon_2)$-block-encoding of A, with a single use of W, P_r and $P_L^\dagger$.*

Proof. Observe that $\tilde{W} = (P_L^\dagger \otimes I_a \otimes I_s)W(P_R \otimes I_a \otimes I_s)$ is an $(\alpha\beta, a+b, \alpha\epsilon_1 + \alpha\beta\epsilon_2)$block-encoding of A.

$$\begin{aligned}
\|A - \alpha\beta(\langle 0|^{\otimes b} \otimes \langle 0|^{\otimes a} \otimes I)\tilde{W}(|0\rangle^{\otimes b} \otimes |0\rangle^{\otimes a} \otimes I)\| &= \|A - \alpha \sum_{j=0}^{m-1} \beta(c_j^* d_j)(\langle 0|^{\otimes a} \otimes I)U_j(|0\rangle^{\otimes a} \otimes I)\| \\
&\leq \alpha\epsilon_1 + \|A - \alpha \sum_{j=0}^{m-1} y_j(\langle 0|^{\otimes a} \otimes I)U_j(|0\rangle^{\otimes a} \otimes I)\| \\
&\leq \alpha\epsilon_1 + \alpha \sum_{j=0}^{m-1} y_j \|A_j - (\langle 0|^{\otimes a} \otimes I)U_j(|0\rangle^{\otimes a} \otimes I)\| \\
&\leq \alpha\epsilon_1 + \alpha \sum_{j=0}^{m-1} y_j \epsilon_2 \\
&\leq \alpha\epsilon_1 + \alpha\beta\epsilon_2
\end{aligned}$$

□

2.3 Examples

Linear combinations of block-encodings are used to obtain mixed-parity functions in QSVT required for Hamiltonian simulation. LCU trick is used for Hamiltonian simulation, or to instantiate block-encodings of chemistry or condensed matter physics Hamiltonians.

Example 2.3.1. (Linear combination of two matrices). Let U_A, U_B be two n-qubit unitaries, and we would like to construct a block encoding of $T = U_A + U_B$.

There are two terms in total, so one ancilla qubit is needed. The prepare oracle needs to implement

$$V|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

so this is the Hadamard gate. The circuit is given by figure 2.2, which constructs $W \in BE_{\sqrt{2},1}(T)$. A special case is the linear combination of two block encoded

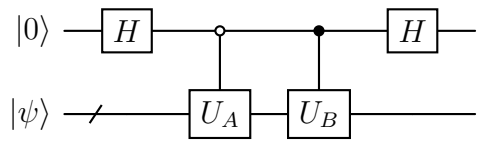


Figure 2.2: Circuit for linear combination of two unitaries

matrices. Given two n-qubit matrices A, B, for simplicity let $U_A \in BE_{1,m}(A), U_B \in BE_{1,m}(B)$. We would like to construct a block encoding of $T = A + B$. The circuit is given by figure 2.3, which constructs $W \in BE_{2,m+1}(T)$. This is also an example of a linear combination of non-unitary matrices.

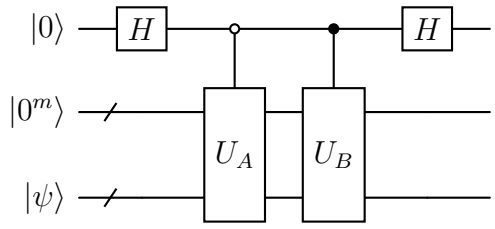


Figure 2.3: Circuit for linear combination of two block encoded matrices

Example 2.3.2. (Transverse field Ising Model) Consider the following TFIM model with periodic boundary conditions ($Z_n = Z_0$) and $n = 2^m$.

$$\hat{H} = -\sum_{i\in[n]} Z_i Z_{i+1} - \sum_{i\in[n]} X_i$$

In order to use LCU we need (m+1) ancilla qubits. The prepare oracle can be simply constructed from the Hadamard gate

$$V = H^{\otimes(m+1)}$$

and the select oracle implements

$$U = \sum_{i\in[n]} |i\rangle\langle i| \otimes (-Z_i Z_{i+1} + \sum_{i\in[n]} |i+n\rangle\langle i+n| \otimes (-X_i)$$

The corresponding $W \in BE_{\sqrt{2n},n+1}(\hat{H})$.

Example 2.3.3. (Block encoding of a matrix polynomial) Let us use the LCU lemma to construct the block encoding for an arbitrary matrix polynomial for a Hermitian matrix A.

$$f(A) = \sum_{k\in[K]} \alpha_k T_k(A)$$

with $\|\alpha\|_1 = \sum_{k\in[K]} |\alpha_k|$ and we set $K = 2^a$. For simplicity assume $\alpha_k \geq 0$.

We have constructed $U_k = (U_A Z_\Pi)^k$ as the $(1, m)$ block encoding of $T_k(A)$. From each U_k we can implement the select oracle

$$U = \sum_{k\in[K]} |k\rangle \langle k| \otimes U_k$$

via multi-qubit controls. Also given the availability of the prepare oracle

$$V|0^a\rangle = \frac{1}{\sqrt{\|\alpha\|_1}} \sum_{k\in[K]} \sqrt{\alpha_k}\,|k\rangle$$

we obtain a $(\|\alpha\|_1, m + a)$-block-encoding of $f(A)$.

The need of using a ancilla qubtis, and even more importantly the need to implement the prepare and select oracles is undesirable. We will later see that the quantum signal processing (QSP) and quantum singular value transformation (QSVT) can drastically reduce bot sources of difficulties.

Example 2.3.4. (Matrix functions given by a matrix Fourier Series). Instead of block encoding, LCU can also utilize a different query model based on Hamiltonian simulation. Let A be an n-qubit Hermitian matrix. Consider $f(x) \in \mathbb{R}$ given by its Fourier expansion (up to a normalization factor)

$$f(x) = \int \hat{f}(x) e^{\iota k x} dk$$

and we are interested in computing the matrix function via numerical quadrature

$$f(A) = \int \hat{f}(k) e^{\iota k A} dk \approx \Delta k \sum_{k\in[K]} \hat{f}(k) e^{\iota k A}$$

Here K is a uniform grid discertizing the interval $[-L, L]$ using $|K| = 2^p$ grid points, and the grid space is $\Delta k = 2L/|K|$. The prepare oracle is given by the coefficients $c_k = \Delta k \hat{f}(k)$, and the corresponding subnormalization factor is

$$\|c\|_1 = \sum_{k\in[K]} \Delta k |\hat{f}(k)| \approx \int |\hat{f}(k)| dk$$

The select oracle is

$$U = \sum_{k \in K} |k\rangle \langle k| \otimes e^{\iota k A}$$

This can be efficiently implemented using the controlled matrix powers using controlled rotations, where the basic unit is the short time Hamiltonian simulation $e^{\iota \Delta k A}$. This can be used to block encode a large class of matrix functions.

Important Note

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. Given a single block-encoded operator A, the above primitives can be used to implement a block-encoding polynomial in A. However, this can be achieved using quantum singular value transformation.

Chapter 3

Matrix functions of Hermitian Matrices

Let A be an n-qubit Hermitian matrix. Then A has the eigenvalue decomposition

$$A = V\Lambda V^\dagger$$

Here $\Lambda = diag(\{\lambda_i\})$ is a diagonal matrix, and $\lambda_0 \leq \ldots \leq \lambda_{N-1}$. Let the scalar function f be well defined on all λ_i's. then the matrix function $f(A)$ can be defined on the eigendecomposition:

Definition 3.0.1. (Matrix functions of Hermitian matrices) Let $A \in \mathbb{C}^{N\times N}$ be a Hermitian matrix with eigenvalue decomposition $V\Lambda V^\dagger$. Let $f : \mathbb{R} \to \mathbb{C}$ be a scalar function such that $f(\lambda_i)$ is defined for all $i \in [N]$. The matrix function is defined as

$$f(A) = Vf(\Lambda)V^\dagger$$

where

$$f(\Lambda) = diag(f(\lambda_0), f(\lambda_1), \ldots, f(\lambda_{N-1}))$$

This chapter introduces techniques to construct an efficient quantum circuit to compute $f(A)\,|b\rangle$ for any state $|b\rangle$. Throughout the discussion we assume A is queried in the block encoding model denoted by U_A. For simplicity we assume that there is no error in the block encoding, i.e., $U_A \in BE_{\alpha,m}(A)$, and without the loss of generality we can take $\alpha = 1$.

Many tasks in scientific computation can be expressed in terms of matrix functions. Here are a few examples:

- Hamiltonian simulation: $f(A) = e^{\iota At}$

- Gibbs state preparation: $f(A) = e^{-\beta A}$
- Solving linear systems of equation $f(A) = A^{-1}$
- Eigenstate filtering $f(A) = \mathbb{I}_{(-\infty,0)}(A - \mu I)$

A key technique for representing matrix functions is called qubitization.

Chapter 4

Qubitization

4.1 Motivation

Qubitization has the following motivation: we are given block-encoding of U_A of a Hermitian operator A, and we wish to manipulate A - e.g., implement A^2, or more generally some function $f(A)$.However, the eigenvalues of U_A are typically unrelated to those of A, and plain repeated applications of U_A do not in general produced 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:

1. W is a block-encoding of the operator A.
2. The spectrum of W has a nice relation to the spectrum of A.
3. Repeated applications 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 be implemented by using quantum signal processing.

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, providing a potentially cheaper alternative to such tasks compared to approaches based on explicit Hamiltonian simulation for implementing $U = e^{\iota At}$.

4.2 Qubitization of Hermitian matrices with Hermitian block encoding

We first introduce some heuristic idea behind qubitization. For any $-1 < \lambda \leq 1$, we can consider a 2×2 rotation matrix,

$$O(\lambda) = \begin{pmatrix} \lambda & -\sqrt{1-\lambda^2} \\ \sqrt{1-\lambda^2} & \lambda \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

where we have performed the change of variable $\lambda = \cos\theta$ with $0 \leq \theta < \pi$

Now direct computations show that

$$O^k(\lambda) = \begin{pmatrix} \cos(k\theta) & -\sin(k\theta) \\ \sin(k\theta) & \cos(k\theta) \end{pmatrix}$$

Using the definition of Chebyshev Polynomials (of first and second kinds, respectively)

$$T_k(\lambda) = \cos\left(k \cos^{-1}\lambda\right), \quad U_{k-1}(\lambda) = \frac{\sin k\theta}{\sin\theta} = \frac{\sin(k\cos^{-1}\lambda)}{\sqrt{1-\lambda^2}}$$

we have

$$O^k(\lambda) = \begin{pmatrix} T_k(\lambda) & -\sqrt{1-\lambda^2}U_{k-1}(\lambda) \\ \sqrt{1-\lambda^2}U_{k-1}(\lambda) & T_k(\lambda) \end{pmatrix}$$

Note that if we somehow replace λ by A, we have immediately obtain a $(1,1)$ block encoding for the Chebyhsev polynomial $T_k(A)$! This is precisely what qubitization aims at achieving, though there are some small twists.

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

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

where $|0^m\rangle$ denotes $|0\rangle^{\otimes m}$. In the simplest scenario we assume that $U_A \in HBE_{1,m}(A)$. Start from the spectral decomposition

$$A = \sum_i \lambda_i |v_i\rangle \langle v_i|$$

we have that for each eigenstate $|v_i\rangle$,

$$U_A |0^m\rangle |v_i\rangle = |0^m\rangle A |v_i\rangle + |\tilde{\perp}_i\rangle = \lambda_i |0^m\rangle |v_i\rangle + |\tilde{\perp}_i\rangle$$

Here $|\tilde{\perp}_i\rangle$ is an unnormalized state that is orthogonal to all states of the form $|0^m\rangle|\psi\rangle$, i.e.,

$$\Pi|\tilde{\perp}\rangle = 0$$

where

$$\Pi = |0^m\rangle\langle 0^m| \otimes I_n$$

is a projection operator.

Since the right hand side of equation is an unnomralized state, we may also write

$$|\tilde{\perp}_i\rangle = \sqrt{1-\lambda_i^2}\,|\perp_i\rangle$$

where $|\perp_i\rangle$ is a normalized state.

Now if $\lambda_i = \pm 1$, then $\mathcal{H}_i = span\{|0^m\rangle|v_i\rangle\}$ is already an invariant subspace of U_A, and $|\perp_i\rangle$ can be any state. Otherwise, use the fact that $U_A = U_A^\dagger$, we can apply U_A again to both sides of the equation and obtain

$$\begin{aligned}
U_A^2|0^m\rangle|v_i\rangle &= U_A(\lambda_i|0^m\rangle|v_i\rangle + |\tilde{\perp}_i\rangle)\\
|0^m\rangle|v_i\rangle &= \lambda_i U_A|0^m\rangle|v_i\rangle + U_A\sqrt{1-\lambda_i^2}\,|\perp_i\rangle\\
&= \lambda_i(\lambda_i|0^m\rangle|v_i\rangle + |\tilde{\perp}_i\rangle) + \sqrt{1-\lambda_i^2}U_A|\perp_i\rangle\\
&= \lambda_i^2|0^m\rangle|v_i\rangle + \lambda_i\sqrt{1-\lambda_i^2}\,|\perp_i\rangle + \sqrt{1-\lambda_i^2}U_A|\perp_i\rangle\\
|0^m\rangle|v_i\rangle - \lambda_i^2|0^m\rangle|v_i\rangle &= \sqrt{1-\lambda_i^2}(\lambda_i|\perp_i\rangle + U_A|\perp_i\rangle)\\
\frac{1-\lambda_i^2}{\sqrt{1-\lambda_i^2}}|0^m\rangle|v_i\rangle &= \lambda_i|\perp_i\rangle + U_A|\perp_i\rangle\\
\implies U_A|\perp_i\rangle &= \sqrt{1-\lambda_i^2}\,|0^m\rangle|v_i\rangle - \lambda_i|\perp_i\rangle
\end{aligned}$$

Therefore $\mathcal{H}_i = span\{|0^m\rangle|v_i\rangle, |\perp_i\rangle\}$ is an invariant subspace of U_A. Furthermore, the matrix representation of U_A with respect to the basis $\mathcal{B}_i = \{|0^m\rangle|v_i\rangle, |\perp_i\rangle\}$ is

$$[U_A]_{\mathcal{B}_i} = \begin{pmatrix} \lambda_i & \sqrt{1-\lambda_i^2} \\ \sqrt{1-\lambda_i^2} & -\lambda_i \end{pmatrix}$$

i.e., U_A restricted to $\mathcal{H}_i$ is a reflection operator. This also leads to the name "qubitization", which means that each eigenvector $|v_i\rangle$ is "qubitized" into a two-dimensional space $\mathcal{H}_i$.

In order to construct a block encoding for $T_k(A)$, we need to turn U_A into a rotation. For this note that $\mathcal{H}_i$ is also an invariant subspace for the projection operator Π:

$$\Pi \left|0^m\right\rangle \left|v_i\right\rangle = (\left|0^m\right\rangle \left\langle 0^m\right| \otimes I_n)(\left|0^m\right\rangle \otimes \left|v_i\right\rangle) = \left|0^m\right\rangle \otimes \left|v_i\right\rangle = \left|0^m\right\rangle \left|v_i\right\rangle$$

and

$$\Pi \left|\tilde{\perp}_i\right\rangle = 0$$

Thus, combining the results that $\Pi \left|0^m\right\rangle \left|v_i\right\rangle = \left|0^m\right\rangle \left|v_i\right\rangle$ and $\Pi \left|\perp_i\right\rangle = 0$, we get,

$$[\Pi]_{\mathcal{B}_i} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Similarly define $Z_\Pi = 2\Pi - 1$, since

$$[Z_\Pi]_{\mathcal{B}_i} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Z_Π acts as a reflection operator restricted to each subspace $\mathcal{H}_i$. Then $\mathcal{H}_i$ is the invariant subspace for the iterate

$$O = U_A Z_\Pi$$

and

$$[O]_{\mathcal{B}_i} = \begin{pmatrix} \lambda_i & -\sqrt{1-\lambda_i^2} \\ \sqrt{1-\lambda_i^2} & \lambda_i \end{pmatrix}$$

is the desired rotation matrix. Therefore

$$[O^k]_{\mathcal{B}_i} = [(U_A Z_\Pi)^k]_{\mathcal{B}_i} = \begin{pmatrix} T_k(\lambda_i) & -\sqrt{1-\lambda_i^2}U_{k-1}(\lambda_i) \\ \sqrt{1-\lambda_i^2}U_{k-1}(\lambda_i) & T_k(\lambda_i) \end{pmatrix}$$

Since $\{\left|0^m\right\rangle \left|v_i\right\rangle\}$ spans the range of Π, we have

$$O^k = \begin{pmatrix} T_k(A) & * \\ * & * \end{pmatrix}$$

i.e., $O^k = (U_A Z_\Pi)^k$ is a $(1, m)$-block encoding of the Chebyshev polynomial $T_k(A)$.

In order to implement Z_Π, note that if $m = 1$, then Z_Π is just the Pauli Z gate. When $m > 1$, the circuit returns $\left|1\right\rangle \left|0^m\right\rangle$ if $b = 0^m$, and $-\left|1\right\rangle \left|b\right\rangle$ if $b \neq 0^m$. So this precisely implements Z_Π where the signal qubit $\left|1\right\rangle$ is used as a work register. We may also discard the signal qubit, and resulting unitary is denoted by Z_Π.

In other words, the circuit in figure implements the operator O. Repeating the circuit k times gives the $(1, m + 1)$-block encoding of $T_k(A)$.

Remark. **(Alternative perspective of qubitization)** The fact that an arbitrarily large block encoding matrix U_A can be partially block diagonalized into N subblocks of size 2×2 may seem a rather peculiar algebraic structure. In fact there are other alternative perspectives and derivations of the qubitization result. Some noticeable ones include the use of Jordan's Lemma, and the use of cosine-sine (CS) decomposition.

4.3 Qubitization of hermitian matrices with general block encoding

Previously we assumed that $U_A = U_A^\dagger$ to block encode a Hermitian matrix A. For instance, a sparse Hermitian matrices, such Hermitian block encoding can be constructed following the recipe as done in the previous section. However, this can come at the expense of requiring additional structures and oracles. In general, the block encoding of a Hermitian matrix may not be hermitian itself. In this section we demonstrate that the strategy of qubitization can be modified to accommodate general block encodings.

Again start from the eigen decomposition $A = \sum_i \lambda_i |\lambda_i\rangle \langle\lambda_i|$, we apply U_A to $|0^m\rangle |v_i\rangle$ and obtain

$$U_A |0^m\rangle |v_i\rangle = \lambda_i |0^m\rangle |v_i\rangle + \sqrt{1-\lambda_i^2}\, |\perp_i'\rangle$$

where $|\perp_i\rangle$ is a normalized state satisfying $\Pi |\perp_i'\rangle = 0$

Since U_A block encodes a Hermitian matrix A, we have

$$U_A^\dagger = \begin{pmatrix} A & * \\ * & * \end{pmatrix}$$

which implies that there exist another normalized state $|\perp_i\rangle$ satisfying $\Pi |\perp_i\rangle = 0$ and

$$U_A^\dagger |0^m\rangle |v_i\rangle = \lambda_i |0^m\rangle |v_i\rangle + \sqrt{1-\lambda_i^2}\, |\perp_i\rangle$$

Now apply U_A to both sides, we obtain

$$|0^m\rangle |v_i\rangle = \lambda_i^2 |0^m\rangle |v_i\rangle + \lambda_i \sqrt{1-\lambda_i^2}\, |\perp_i'\rangle + \sqrt{1-\lambda_i^2} U_A |\perp_i\rangle$$

which gives

$$U_A |\perp_i\rangle = \sqrt{1-\lambda_i^2}\, |0^m\rangle |v_i\rangle - \lambda_i |\perp_i'\rangle$$

Define

$$\mathcal{B}_i = \{|0^m\rangle |v_i\rangle, |\perp_i\rangle\}, \quad \mathcal{B}_i' = \{|0^m\rangle |v_i\rangle, |\perp_i'\rangle\}$$

and the associated two-dimensional subspaces $\mathcal{H}_i = span(\mathcal{B}_i)$, $\mathcal{H}_i' = span(\mathcal{B}_i')$, we find that U_A maps $\mathcal{H}_i$ to $\mathcal{H}_i'$. Correspondingly $U_A^\dagger$ maps $\mathcal{H}_i'$ to $\mathcal{H}_i$.

Then in the matrix representation we get,

$$[U_A]_{\mathcal{B}_i}^{\mathcal{B}_i'} = \begin{pmatrix} \lambda_i & \sqrt{1-\lambda_i} \\ \sqrt{1-\lambda_i^2} & -\lambda_i \end{pmatrix}$$

Similar calculations show that

$$[U_A^\dagger]_{\mathcal{B}_i'}^{\mathcal{B}_i} = \begin{pmatrix} \lambda_i & \sqrt{1-\lambda_i^2} \\ \sqrt{1-\lambda_i^2} & -\lambda_i \end{pmatrix}$$

Meanwhile both $\mathcal{H}_i$ and $\mathcal{H}_i'$ are the invariant subspace of the projector Π, with matrix representation

$$[\Pi]_{\mathcal{B}_i} = [\Pi]_{\mathcal{B}_i'} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Therefore

$$[Z_\Pi]_{\mathcal{B}_i} = [Z_\Pi]_{\mathcal{B}_i'} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Hence $\mathcal{H}_i$ is an invariant subspace of $\tilde{O} = U_A^\dagger Z_\Pi U_A Z_\Pi$, with matrix representation

$$[\tilde{O}]_{\mathcal{B}_i} = \begin{pmatrix} \lambda_i & -\sqrt{1-\lambda_i^2} \\ \sqrt{1-\lambda_i^2} & \lambda_i \end{pmatrix}^2$$

repeating k times, we have

$$[\tilde{O}^k]_{\mathcal{B}_i} = (U_A^\dagger Z_\Pi U_A Z_\Pi)^k = \begin{pmatrix} \lambda_i & -\sqrt{1-\lambda_i^2} \\ \sqrt{1-\lambda_i^2} & \lambda_i \end{pmatrix}^{2k}$$

Since any vector $|0^m\rangle |\psi\rangle$ can be expanded in terms of the eigenvectors $|0^m\rangle |v_i\rangle$, we have

$$(U_A^\dagger Z_\Pi U_A Z_\Pi)^k = \begin{pmatrix} T_{2k}(A) & * \\ * & * \end{pmatrix}$$

Therefore if we would like to construct an even order Chebyshev polynomial $T_{2k}(A)$, the circuit $(U_A Z_\Pi U_A Z_\Pi)^k$ straightforwardly gives a $(1, m)$-block-encoding.

In order to construct the block-encoding of an odd polynomial $T_{2k+1}(A)$, we note that

$$[U_A Z_\Pi (U_A^\dagger Z_\Pi U_A Z_\Pi)^k]^{\mathcal{B}_i'}_{\mathcal{B}_i} = \begin{pmatrix} T_{2k+1}(\lambda_i) & -\sqrt{1-\lambda_i^2}U_{2k}(\lambda_i) \\ \sqrt{1-\lambda_i^2}U_{2k}(\lambda_i) & T_{2k+1}(\lambda_i) \end{pmatrix}$$

Using the fact that $\mathcal{B}_i, \mathcal{B}_i'$ share the common basis $|0^m\rangle |v_i\rangle$, we still have the block-encoding

$$U_A Z_\Pi (U_A^\dagger Z_\Pi U_A Z_\Pi)^k = \begin{pmatrix} T_{2k+1}(A) & * \\ * & * \end{pmatrix}$$

Therefore $U_A Z_\Pi (U_A^\dagger Z_\Pi U_A Z_\Pi)^k$ is a $(1, m)$-block encoding of $T_{2k+1}(A)$.

In summary, the block encoding of $T_l(A0$ is given by applying $U_A Z_\Pi$ and $U_A^\dagger Z_\Pi$ alternatively. If $l = 2k$, then there are exactly k such pairs. The quantum circuit for each pair $\tilde{O}$ is

Otherwise if $l = 2k + 1$, then there is an extra $U_A Z_\Pi$. The effect is to map each eigenvector $|v\rangle |v_i\rangle$ back and forth between the two-dimensional subspaces $\mathcal{H}_i$ and $\mathcal{H}_i'$. We shall later see that the separate treatment even/odd polynomials will play a more prominent role.

Now that we have obtained $O_k \in BE_{1,m}(T_k(A))$ for all k, we can use the LCU again to construct block encodings for linear combination of Chebyshev polynomials.

4.4 Dominant resource cost - qubits and gates

The resource cost of qubitization is inheritied from the cost of the block-encoding. Given a Hermitian $(\alpha, m, 0)$-block-encoding, the qubitization operator @ 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 controlled Z gate, equivalent to an m-qubit Toffoli gate up to single-qubit gates. An example qubitization circuit is shown below in figure 4.1 for $m = 3$. Implementing a block-encoding of a degree d Chebyshev polynomial applied to A requires d calls to U_A and $Z_{|0^m\rangle}$. if the block-encoding of U_A is not Hermitian, qubitization can be achieved using the construction that uses one additional qubit and one call to controlled U_A and controlled $U_A^\dagger$ to implement the Hermitian block-encoding.

$$U_A' = ((HX) \otimes I)(|0\rangle \langle 0| \otimes U_A + |1\rangle \langle 1| \otimes U_A^{|dagger})(H \otimes I)$$

An alternate to qubitization is based on singular value transformation 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

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

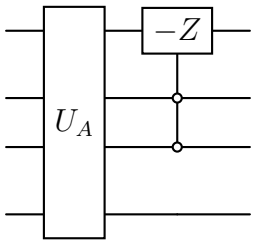


Figure 4.1: An example of qubitization circuit using the Hermitian $(1, 3, 0)$ -block-encoding of U_A

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

Important Note

The original formualtion of qubitization discussed above requires a Hermitian or normal block-encoded matrix A. The concept can be extended to general (non-square) matrices via quantum singular value transformation, providing a significant generalizatin, however in some cases quantum signal processing and its generalized versions can exploit additional structure that comes for example from the extra symmetries of Hermitian block-encoding leading to potentia cost factor savings.
Consider for example Hamiltonian simulation, where QSVT separately implements $\sin(tH)$ and $\cos(tH)$ using a block-encoding of U_H of the Hamiltonian H, and applies a 3 step oblivious amplification procedure on top of linear combination of unitaries to implement $exp(\iota tH)$. Meanwhile, quantum signal processing implements $exp(\iota tH)$ directly but requires an additional ancilla qubit and controlled access to a Hermitian block encoding $U_H^\dagger$, which, when implemented, 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.

4.5 Example - Use cases

- Some quantum algorithms in quantum chemistry that compute energies perform phase estimation on a qubiziation 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 $e^{\iota Ht}$, implemented via Hamiltonian simulation.

- Qubitization acts as a precursor to quantum singular value transformation, which extends the concept to general matrices and unifies it with quantum signal processing.

Chapter 5

Quantum Eigenvalue Transformation

We know how to construct a block encoding of an Hermitian matrix A (the block encoding matrix itself can be non-Hermitian), use qubitization to block encode $T_k(A)$, use LCU to block encode an arbitrary polynomial function of A (up to a subnormalization factor). This framework is conceptually appearing, but the practical implementation of the select and prepare oracles are by no means straightforward, and can come at significant costs.

5.1 Hermitian Block encoding

Note that,

$$\iota Z = \iota \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} \iota & 0 \\ 0 & -\iota \end{bmatrix}$$

Also,

$$R_Z(\theta) = e^{-\iota \frac{\theta}{2} Z} = \begin{bmatrix} e^{-\iota\theta/2} & 0 \\ 0 & e^{\iota\theta/2} \end{bmatrix} \implies e^{\iota \frac{\pi}{2} Z} = \begin{bmatrix} e^{\iota\pi/2} & 0 \\ 0 & e^{-\iota\pi/2} \end{bmatrix} = \begin{bmatrix} \iota & 0 \\ 0 & -\iota \end{bmatrix}$$

Thus, $\iota Z = e^{\iota \frac{\pi}{2} Z}$, which implies that $Z = -\iota e^{\iota \frac{\pi}{2} Z}$. If A is given by a Hermitian block encoding U_A, the block encoding of the Chebyshev polynomial can be written

$$\begin{aligned} O^d &= (U_A Z_\Pi)^d \\ &= (-\iota U_A e^{\iota \frac{\pi}{2} Z})^d \\ &= (-\iota)^d (U_A e^{\iota \frac{\pi}{2} Z})^d \\ &= (-\iota)^d \prod_{j=1}^{d} (U_A e^{\iota \frac{\pi}{2} Z_\Pi}) \end{aligned}$$

This is a special case of the following representation. Note that $(-\iota)^d$ is an irrelevant phase factor and can be discarded.

$$U_{\tilde{\Phi}} = e^{\iota \tilde{\phi}_0 Z_\Pi} \prod_{j=1}^{d} (U_A e^{\iota \tilde{\phi}_j Z_\Pi})$$

The representation is called a Quantum Eigenvalue Transformation (QET).

Due to qubitization, $U_{\tilde{\phi}}$ should block encode some matrix polynomial of A. We first state the following theorem without proof.

Theorem 5.1.1. ***(Quantum signal processing, a simplified version)*** *Consider*

$$U_A = \begin{pmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{pmatrix}$$

For any $\tilde{\Phi} = (\tilde{\phi}_0, \ldots, \tilde{\phi}_d) \in \mathbb{R}^{d+1}$,

$$U_{\tilde{\Phi}} = e^{\iota \tilde{\phi}_0 Z} \prod_{j=1}^{d} (U_A e^{\iota \tilde{\phi}_j Z}) = \begin{pmatrix} P(x) & * \\ * & * \end{pmatrix}$$

where $P \in \mathbb{C}[x]$ *satisfy*

1. $deg(P) \leq d$
2. P *has parity* $(d \bmod 2)$
3. $|P(x)| \leq 1 \forall x \in [-1, 1]$.

Also define $-\tilde{\Phi} = (-\tilde{\phi}_0, \ldots, -\tilde{\phi}_d) \in \mathbb{R}^{d+1}$, *then*

$$U_{-\tilde{\Phi}} = e^{-\iota\tilde{\phi}_j} Z \prod_{j=1}^{d} (U_A e^{-\iota\tilde{\phi}_j Z}) = U_{\tilde{\phi}}^* = \begin{pmatrix} P^*(x) & * \\ * & * \end{pmatrix}$$

Here $P^*(x)$ *is the complex conjugation of* $P(x)$, *and* $U_{\tilde{\Phi}}^*$ *is the complex conjugation (without transpose) of* $U_{\tilde{\Phi}}$.

Remark. This theorem can be proved inductively. However, this is a special case of the quantum signal processing, so we will omit the proof here. In fact, we will also state the converse of the result, which describes precisely the class of matrix polynomials that can be described by such phase factor modulations. The condition (1) states that the polynomial degree is upper bounded by the number of U_A's and the condition (3) is simply a consequence of that $U_{\tilde{\phi}}$ is a unitary matrix. The condition (2) is less obvious, but should not come at a surprise, since we have seen the need of treating even and odd polynomials separately in the case of qubitization with a general block encoding can be proved directly by taking the complex conjugation of $U_{\tilde{\Phi}}$.

Following the qubitization procedure we immediately have the following theorem.

Theorem 5.1.2. ***(Quantum Eigenvalue transformation with Hermitian block encoding)*** *Let* $U_A \in HBE_{1,m}(A)$. *Then for any* $\tilde{\Phi} = (\tilde{\phi}_0, \ldots, \tilde{\phi}_d) \in \mathbb{R}^{d+1}$,

$$U_{\tilde{\Phi}} = e^{\iota\tilde{\Phi}_0 Z_\Pi} \prod_{j=1}^{d} (U_A e^{\iota\tilde{\Phi}_j Z_\Pi}) = \begin{pmatrix} P(A) & * \\ * & * \end{pmatrix} \in BE_{1,m}(P(A))$$

where $P \in \mathbb{C}[x]$ *satisfies the requirements in the theorem 5.1.1.*

Using this Theorem, we may construct the block encoding of a matrix polynomial without invoking LCU. The cost is essentially the same as block encoding a Chebyshev polynomial.

In order to implement $e^{\iota\phi Z_\Pi}$, we note that the quantum circuit denoted by CR_ϕ is in figure 5.1 returns $e^{\iota\phi} |0\rangle |0^m\rangle$ if $b = 0^m$, and $e^{-\iota\phi} |0\rangle |b\rangle$ if $b \neq 0^m$. So omitting the signal qubit, this is precisely $e^{\iota\phi Z_\Pi}$. Therefore, if A is given by a Hermitian block encoding U_A, we can follow the argument in previous sections and construct the following unitary. The corresponding quantum circuit is in figure 5.2, which uses one extra ancilla qubit. When measuring the $(m+1)$-ancilla qubits and obtain $|0\rangle |0^m\rangle$ the corresponding (unnormalized) state in the system register is $P(A) |\psi\rangle$. The QET described by the circuit in figure 5.2 generally construct a block encoding of $P(A)$

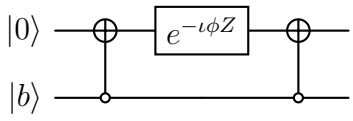


Figure 5.1: Implementing the controlled rotation circuit for quantum eigenvalue transformation

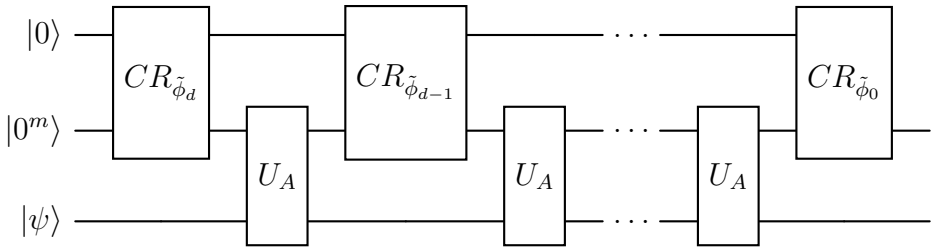
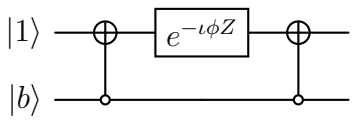


Figure 5.2: Circuit of quantum eigenvalue transformation to construct $U_{P(A)} \in BE_{1,m+1}(P(A))$, using $U_A \in HBE_{1,m}(A)$

for some complex polynomial P. In practical applications (such as those later in this chapter) we would like to construct a block encoding of $P_{Re}(A) = (ReP)(A) = \frac{1}{2}(P(A) + P^*(A))$ instead. Below we demonstrate that a simple modification of figure 5.2 allows us to achieve this goal. To this end, we use equation

$$U_{-\tilde{\Phi}} = e^{-\iota\tilde{\Phi}_0 Z_\Pi} \prod_{j=1}^{d} (U_A e^{-\iota\tilde{\Phi}_j Z_\Pi}) = \begin{pmatrix} P^*(A) & * \\ * & * \end{pmatrix}$$

Qubitization allows us to construct. So all we need is to negate all phase factors in $\tilde{\Phi}$. In order to implement $CR_{-\phi}$, we do not actually need to implement a new circuit. Instead we may simply change the signal qubit from $|0\rangle$ to $|1\rangle$. Now we claim the



circuit in figure 5.3 implements a block encoding $U_{P_{Re(A)}} \in BE_{1,m+1}(P_{Re}(A))$. This circuit can be viewed as an implementation of the linear combination of unitaries $\frac{1}{2}(U_{P^*(A)} + U_{P(A)})$. To verify this, we may evaluate

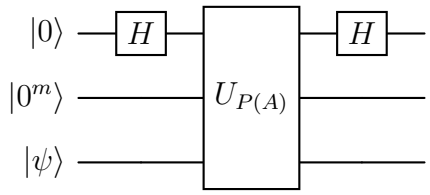


Figure 5.3: Circuit of quantum eigenvalue transformation for constructing a $(1, m+1)$-block encoding of $P_{Re}(A)$

$$
\begin{aligned}
|0\rangle |0^m\rangle |\psi\rangle &\xrightarrow{H\otimes I_{m+n}} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) |0^m\rangle |\psi\rangle \\
&\xrightarrow{U_{P(A)}} \frac{1}{\sqrt{2}} |0\rangle (|0^m\rangle P(A) |\psi\rangle + |\perp\rangle) + \frac{1}{\sqrt{2}} |1\rangle (|0^m\rangle P^*(A) |\psi\rangle + |\perp'\rangle) \\
&\xrightarrow{H\otimes I_{m+n}} |0\rangle \left(|0^m\rangle \frac{P(A) + P^*(A)}{2} |\psi\rangle \right) + |\tilde{\perp}\rangle \\
&= |0\rangle |0^m\rangle P_{Re}(A) |\psi\rangle + |\tilde{\perp}\rangle
\end{aligned}
$$

here $|\perp\rangle$ and $|\perp'\rangle$ are two $(m+n)$-qubit state orthogonal to any state $|0^m\rangle |x\rangle$, while $|\tilde{\perp}\rangle$ is a $(m+n+1)$- qubit state orthogonal to any state $|0\rangle |0^m\rangle |x\rangle$. In other words, by measuring all $(m+1)$ ancilla qubits and obtain 0^{m+1}, the corresponding (unnormalized) state in the system register is $P_{Re}(A) |\psi\rangle$.

5.1.1 General Block Encoding

If A is given by a general block encoding U_A, the quantum eigenvalue transformation should consist of an alternating sequence of U_A, $U_A^\dagger$ gates. The circuit is given by figure 5.4, and the corresponding block encoding is described in the following theorem. Note that the Hermitian block encoding becomes a special case with $U_A = U_A^\dagger$.

Theorem 5.1.3. ***(Quantum eigenvalue transformation with general block encoding)*** *Let $U_A \in BE_{1,m}(A)$. Then for any $\tilde{\Phi} = (\tilde{\phi}_0, \ldots, \tilde{\phi}_d) \in \mathbb{R}^{d+1}$, let*

$$
U_{\tilde{\Phi}} = e^{\iota\tilde{\phi}_0 Z_\Pi} \prod_{j=1}^{d/2} [U_A^\dagger e^{\iota\tilde{\phi}_{2j-1} Z_\Pi} U_A e^{\iota\tilde{\phi}_{2j+1} Z_\Pi}]
$$

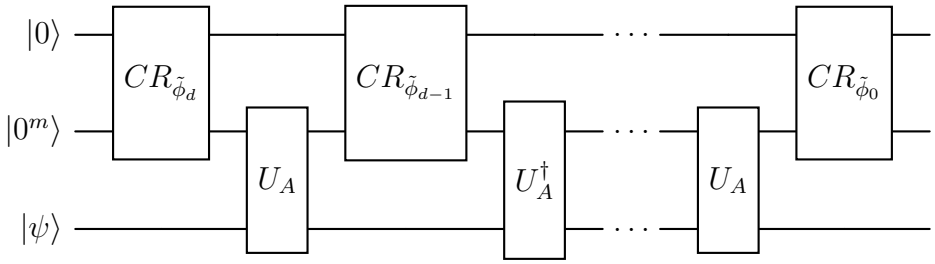


Figure 5.4: Circuit of quantum eigenvalue transformation to construct $U_{P(A)} \in BE_{1,m+1}(P(A))$, using $U_A \in BE_{1,m}(A)$. Here $U_A, U_A^\dagger$ should be applied alternatively. When d is even, the last U_A gate should be replaced by $U_A^\dagger$.

when d is even, and

$$U_{\tilde{\Phi}} = (-\iota)^d e^{\iota\tilde{\phi}_0 Z_\Pi}(U_A e^{\iota\tilde{\Phi}_1 Z_\Pi}) \prod_{j=1}^{(d-1)/2} [U_A^\dagger e^{\iota\tilde{\phi}_{2j} Z_\Pi} U_A e^{\iota\tilde{\phi}_{2j+1} Z_\Pi}]$$

when d is odd. Then

$$U_{\tilde{\Phi}} = \begin{pmatrix} P(A) & * \\ * & * \end{pmatrix} \; inBE_{1,m}(P(A)),$$

where $P \in \mathbb{C}[x]$ satisfy the condition in the theorem 5.1.1.

Following exactly the same procedure, we find that the circuit in figure with $U_{P(A)}$ is given by figure 5.3 implements a $U_{P_{Re}(A)} \in BE_{1,m+1}(P_{Re}(A))$.

5.1.2 General Matrix Polynomials

In practical applications, we may be interested in matrix polynomials $f(A)$, where $f(x) \in \mathbb{R}[x]$ does not have a definite parity. This violates the parity requirement of Theorem. This can be solved using the LCU technique. Note that

$$f(x) = f_{even}(x) + f_{odd}(x)$$

where $f_{even}(x) = \frac{1}{2}(f(x) + f(-x)), f_{odd}(x) = \frac{1}{2}(f(x) - f(-x))$. If $|f(x)| \leq 1$ on $[-1, 1]$, then $|f_{even}(x)|, |f_{odd}(x)| \leq 1$ on $[-1, 1]$, and $f_{even}(x), f_{odd}(x)$ can be each constructed using the circuit in figure. Introducing another ancilla qubit and using the LCU technique we obtain a $(1, m + 1)$-block encoding of $(f_{even}(A) + f_{odd}(A))/2$.

In other words, we obtain a circuit $U_f \in BE_{2,m+2}(f(A))$. Note that unlike the case of block encoding of $P_{Re}(A)$, we lose a subnomalization factor of 2 here. Let us now see how to construct such a block-encoding for a general $f(x) \in \mathbb{R}(x)$ without a definite parity. Since $f_{even}(x)$ and $f_{odd}(x)$ can be easily constructed using the circuit shown in figure 5.3. This can be done because $f^*_{even}(x) = f_{even}(x)$ and similarly $f^*_{odd}(x) = f_{odd}(x)$ since $f(x) \in \mathbb{R}(x)$ and hence both $f_{even}(x), f_{odd}(x) \in \mathbb{R}(x)$. We thus have the block encodings $U_{f_{even}(x)} \in BE_{1,m+1}(f_{even(x)})$ and $U_{f_{odd}(x)} \in BE_{1,m+1}(f_{odd(x)})$. We now wish to have encoding of $f(x) = f_{even}(x) + f_{odd}(x)$ which can be achieved using LCU and an additional qubit. The circuit for this is as shown in the following figure 5.5. This shows that it $U_{f(A)} \in BE_{2,m+2}(f(A))$. To verify this, we may evaluate

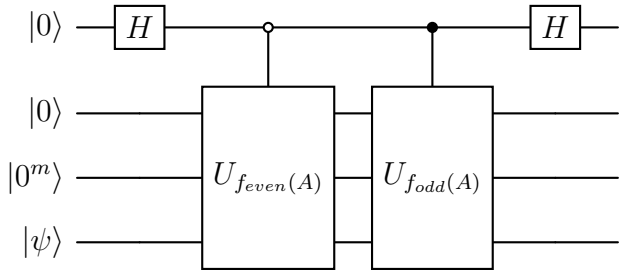


Figure 5.5: Circuit of quantum eigenvalue transformation for constructing a $(2, m+2)$-block encoding of $f(A)$ without definite parity

$$
\begin{aligned}
|0\rangle|0\rangle|0^m\rangle|\psi\rangle &\xrightarrow{H\otimes I_{m+n+1}} \frac{1}{\sqrt{2}}(|0\rangle|0\rangle|0^m\rangle|\psi\rangle + |1\rangle|0\rangle|0^m\rangle|\psi\rangle) \\
&\xrightarrow{U_{f_{even}(A)}} \frac{1}{\sqrt{2}}|0\rangle\left(|0\rangle|0^m\rangle\frac{f_{even}(A)+f^*_{even}(A)}{2}|\psi\rangle + |\tilde{\perp}\rangle\right) + \frac{1}{\sqrt{2}}|1\rangle(|0\rangle|0^m\rangle|\psi\rangle) \\
&\xrightarrow{U_{f_{odd}(A)}} \frac{1}{\sqrt{2}}|0\rangle\left(|0\rangle|0^m\rangle f_{even}(A)|\psi\rangle + |\perp\rangle\right) + \frac{1}{\sqrt{2}}|1\rangle\left(|0\rangle|0^m\rangle f_{odd}(A)|\psi\rangle + |\perp'\rangle\right) \\
&\xrightarrow{H\otimes I_{m+n+1}} |0\rangle\left(|0\rangle|0^m\rangle\frac{f_{even}(A)+f_{odd}(A)}{2}|\psi\rangle\right) + |\tilde{\perp}\rangle \\
&= |0\rangle|0\rangle|0^m\rangle\frac{f(A)}{2}|\psi\rangle + |\tilde{\perp}\rangle
\end{aligned}
$$

Following the same principle, if $f(x) = g(x) + \iota h(x) \in \mathbb{C}[x]$ is a given complex polynomial, and $g, h \in \mathbb{R}[x]$ do not have a definite parity. Here, $g(x) = f(x) + f^*(x)$ and $h(x) = f(x) - f^*(x)$. Now each of $g(x) = g_{even}(x) + g_{odd}(x)$ and $h(x) = h_{even}(x) + h_{odd}(x)$ where $g_{even}(x) = \frac{1}{2}(g(x)+g(-x))$ and $g_{odd}(x) = \frac{1}{2}(g(x)-g(-x))$ and similarly

for $h_{even}(x)$ and $h_{odd}(x)$. Thus, using the same ideas from above, we can construct $U_{g(A)} \in BE_{2,m+2}(g(A)), U_{h(A)} \in BE_{2,m+2}(h(A))$. Then applying another layer of LCU, we obtain $U_{f(A)} \in BE_{4,m+3}(f(A))$. The circuit is as shown in the figure 5.6. To verify this, we may evaluate,

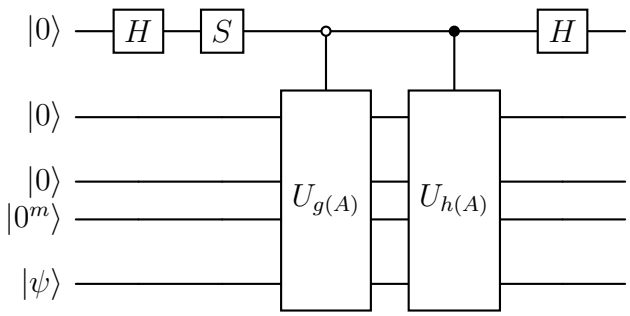


Figure 5.6: Circuit of quantum eigenvalue transformation for constructing a $(4, m+3)$-block encoding of $f(A) = g(A) + \iota h(A)$ where $g, h \in \mathbb{R}$ do not have definite parity

$$
\begin{aligned}
|0\rangle|0\rangle|0\rangle|0^m\rangle|\psi\rangle &\xrightarrow{H\otimes I_{m+n+2}} \frac{1}{\sqrt{2}}(|0\rangle|0\rangle|0\rangle|0^m\rangle|\psi\rangle + |1\rangle|0\rangle|0\rangle|0^m\rangle|\psi\rangle) \\
&\xrightarrow{S\otimes I_{m+n+2}} \frac{1}{\sqrt{2}}(|0\rangle|0\rangle|0\rangle|0^m\rangle|\psi\rangle + \iota|1\rangle|0\rangle|0\rangle|0^m\rangle|\psi\rangle) \\
&\xrightarrow{U_{g(A)}} \frac{1}{\sqrt{2}}|0\rangle(|0\rangle|0\rangle|0^m\rangle\frac{g(A)}{2}|\psi\rangle + |\tilde{\perp}\rangle) + \frac{1}{\sqrt{2}}\iota|1\rangle(|0\rangle|0\rangle|0^m\rangle|\psi\rangle) \\
&\xrightarrow{U_{h(A)}} \frac{1}{\sqrt{2}}|0\rangle(|0\rangle|0\rangle|0^m\rangle\frac{g(A)}{2}|\psi\rangle + |\perp\rangle) + \frac{1}{\sqrt{2}}\iota|1\rangle(|0\rangle|0\rangle|0^m\rangle\frac{h(A)}{2}|\psi\rangle + |\perp'\rangle) \\
&\xrightarrow{H\otimes I_{m+n+2}} |0\rangle\left(|0\rangle|0\rangle|0^m\rangle\frac{g(A)+\iota h(A))}{4}|\psi\rangle\right) + |\tilde{\perp}\rangle \\
&= |0\rangle|0\rangle|0\rangle|0^m\rangle\frac{f(A)}{4}|\psi\rangle + |\tilde{\perp}\rangle
\end{aligned}
$$

On the other hand, if the real and imaginary parts g, h have definite parity, then $U_{g(A)} \in BE_{1,m+1}(g(A)), U_{h(A)} \in BE_{1,m+1}(h(A))$. Applying LCU, we obtain $U_{f(A)} \in BE_{2,m+2}(f(A))$. The corresponding circuit is as shown in the figure 5.7. To verify

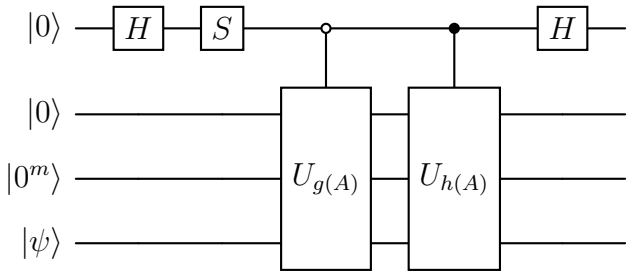


Figure 5.7: Circuit of quantum eigenvalue transformation for constructing a $(2, m+2)$-block encoding of $f(A) = g(A) + \iota h(A)$ where $g, h \in \mathbb{R}$ and have definite parity

this, we may evaluate

$$
\begin{aligned}
|0\rangle |0\rangle |0^m\rangle |\psi\rangle &\xrightarrow{H\otimes I_{m+n+1}} \frac{1}{\sqrt{2}}(|0\rangle |0\rangle |0^m\rangle |\psi\rangle + |1\rangle |0\rangle |0^m\rangle |\psi\rangle) \\
&\xrightarrow{S\otimes I_{m+n+1}} \frac{1}{\sqrt{2}}(|0\rangle |0\rangle |0^m\rangle |\psi\rangle + \iota |1\rangle |0\rangle |0^m\rangle |\psi\rangle) \\
&\xrightarrow{U_{g(A)}} \frac{1}{\sqrt{2}} |0\rangle (|0\rangle |0^m\rangle g(A) |\psi\rangle + |\perp\rangle) + \frac{1}{\sqrt{2}}\iota |1\rangle (|0\rangle |0^m\rangle |\psi\rangle) \\
&\xrightarrow{U_{h(A)}} \frac{1}{\sqrt{2}} |0\rangle (|0\rangle |0^m\rangle g(A) |\psi\rangle + |\perp\rangle) + \frac{1}{\sqrt{2}}\iota |1\rangle (|0\rangle |0^m\rangle h(A) |\psi\rangle + |\perp'\rangle) \\
&\xrightarrow{H\otimes I_{m+n+1}} |0\rangle \left(|0\rangle |0^m\rangle \frac{g(A) + \iota h(A)}{2} |\psi\rangle \right) + |\tilde{\perp}\rangle \\
&= |0\rangle |0\rangle |0^m\rangle \frac{f(A)}{2} |\psi\rangle + |\tilde{\perp}\rangle
\end{aligned}
$$

Chapter 6

Quantum Signal Processing

6.1 Introduction

Quantum signal processing (QSP) describes a method for non linear 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. Remarkably, it has been shown 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 multi-qubit operators that act like single-qubit rotations when restricted to the appropriate two-dimensional subspaces. This insight enables the implementation of block-encoding of polynomials of Hermitian/normal matrices when used in conjunction with qubitization. The two step process of qubitization + QSP can be unified and generalized through quantum singular value transformation (QSVT).

6.1.1 Overview

We follow the "Wx" convention of QSP. We define single-qubit operator

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

which is a single-qubit X rotations. 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 the signal x.

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

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

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

$$U_{QSP}(\Phi) = \begin{pmatrix} P(x) & \iota Q(x)\sqrt{1-x^2} \\ \iota Q^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix}$$

where $P(x), Q(x)$ are complex polynomials obeying a number of constraints, and $P^*(x), Q^*(x)$ denote their complex conjugates.

In terms of implementing matrix polynomials of hermitian matrices, quantum eigenvalue transform provides a much simpler methods than based on the LCU and qubitization (i.e., linear combination of Chebyshev polynomials). The simplification is clear both in terms of the number of ancilla qubits and of the circuit architecture. However, it is not clear so far for which polynomials (either complex polynomial $P \in \mathbb{C}[x]$ or real polynomial $P_{Re} \in \mathbb{R}[x]$) we can apply the QET technique, and how to obtain the phase factors. Quantum signal processing (QSP) provides a complete answer to this question.

Due to qubitization, all these questions can be answered in the context of $SU(2)$ matrices. QSP is the theory of QET for SU(2) matrices, or the unitary representation of a scalar (real or complex) polynomial $P(x)$. Let $A = x \in [-1, 1]$, be a scalar with one-qubit Hermitian block encoding

$$U_A(x) = \begin{pmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{pmatrix}$$

Then

$$O(x) = U_A(x)Z = \begin{pmatrix} x & -\sqrt{1-x^2} \\ \sqrt{1-x^2} & x \end{pmatrix}$$

is a rotation matrix.

The QSP representation takes the following form

$$U(x) = e^{\iota\phi_0 Z}O(x)e^{\iota\phi_1 Z}O(x)\dots e^{\iota\phi_{d-1}Z}O(x)e^{\iota\phi_d Z}$$

By setting $\phi_0 = \ldots = \phi_d = 0$, we immediately obtain the block encoding of the Chebyshev polynomial $T_d(x)$. The representation power of this formulation is characterized in the following theorem. In the following discussion, even functions have parity 0 and odd functions have parity 1.

Theorem 6.1.1. ***(Quantum Signal Processing)*** *There exists a set of phase factors $\Phi = (\phi_0, \dots, \phi_d) \in \mathbb{R}^{d+1}$ such that*

$$U_\Phi(x) = e^{\iota\phi_0 Z}\prod_{j=1}^{d}[O(x)e^{\iota\phi_j Z}] = \begin{pmatrix} P(x) & -Q(x)\sqrt{1-x^2} \\ Q^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix}$$

if and only if $P, Q \in \mathbb{C}[x]$ satisfy

1. *$deg(P) \leq d, deg(Q) \leq d-1$*
2. *P has parity d mod 2 and Q has parity $d-1$ mod 2, and*
3. *$|P(x)|^2 + (1-x^2)|Q(x)|^2 = 1 \forall x \in [-1, 1]$.*

Here $deg(Q) = -1$ means $Q = 0$.

Proof. Let $U_\Phi(x)$ exist with a phase factors $\Phi = (\phi_0, \dots, \phi_d) \in \mathbb{R}^d$. Then we need to show that $P, Q \in \mathbb{C}[x]$ satisfy the given conditions. Since both $e^{\iota\phi Z}$ and $O(x)$ are unitary, the matrix $U_\Phi(x)$ is always a unitary matrix, which immediately implies the condition (3) since the 2-norm of rows of a unitary must sum to 1. Thus,

$$|P(x)|^2 + |-Q(x)\sqrt{1-x^2}|^2 = 1$$

Using the property that $|ab| = |a||b|$ for $a, b \in \mathbb{C}$ and $\forall x \in [-1, 1]$, $|(1-x^2)| \geq 0$, we can simplify the above as follows,

$$|P(x)|^2 + (1-x^2)|Q(x)|^2 = 1$$

Below we only need to verify the conditions (1), (2). This can be done using induction.

When $d = 0$, $U_\Phi(x) = e^{\iota\phi_0 Z}$, which gives $P(x) = e^{\iota\phi_0}$ and $Q = 0$ satisfying all three conditions. $|e^{\iota\phi_0}|^2 = 1$, $deg(e^{\iota\phi_0}) = 0$ which satisfies the condition (2) i.e. the parity is 0 mod 2 $\equiv 0$ (an even function) and $Q = 0$ which corresponds to $deg(Q) = -1$ which also satisfies the parity condition for Q i.e. $0 - 1 \bmod 2 \equiv -1$, which corresponds to $Q = 0$ (both even and odd function). For induction, suppose $U_{(\phi_0,\ldots,\phi_{d-1})}(x)$ takes the form as given in the previous equation with $deg(P) \leq d-1$ and $deg(Q) \leq d-2$, then for any $\phi \in \mathbb{R}$, we have

$$\begin{aligned}
U_{(\phi_0,\ldots,\phi_{d-1},\phi)}(x) &= U_\Phi(x)O(x)e^{\iota\phi Z} \\
&= \begin{pmatrix} P(x) & -Q(x)\sqrt{1-x^2} \\ Q^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix} \begin{pmatrix} x & -\sqrt{1-x^2} \\ \sqrt{1-x^2} & x \end{pmatrix} \begin{pmatrix} e^{\iota\phi} & 0 \\ 0 & e^{-\iota\phi} \end{pmatrix} \\
&= \begin{pmatrix} xP(x) - (1-x^2)Q(x) & -\sqrt{1-x^2}(P(x) + xQ(x)) \\ -\sqrt{1-x^2}(P^*(x) + xQ^*(x)) & xP^*(x) - (1-x^2)Q^*(x) \end{pmatrix} \begin{pmatrix} e^{\iota\phi} & 0 \\ 0 & e^{-\iota\phi} \end{pmatrix} \\
&= \begin{pmatrix} e^{\iota\phi}(xP(x) - (1-x^2)Q(x)) & e^{-\iota\phi}(-\sqrt{1-x^2}(P(x) + xQ(x)) \\ e^{\iota\phi}((-\sqrt{1-x^2}(P^*(x) + xQ^*(x)) & e^{-\iota\phi}(xP^*(x) - (1-x^2)Q^*(x)) \end{pmatrix}
\end{aligned}$$

For the condition (1), note that the new $P'(x)$ can be written as:

$$P'(x) = e^{\iota\phi}(xP(x) - (1-x^2)Q(x))$$

Thus, if $deg(P(x)) \leq d-1 \implies deg(P'(x)) \leq d$ because of the x multiplied. Similarly,

$$\begin{aligned}
-\sqrt{1-x^2}Q'(x) &= e^{-\iota\phi}(-\sqrt{1-x^2}(P(x) + xQ(x)) \\
&= -\sqrt{1-x^2}\left(e^{-\iota\phi}\left(P(x) + \frac{x}{\sqrt{1-x^2}}Q(x)\right)\right) \\
Q'(x) &= e^{-\iota\phi}\left(P(x) + \frac{x}{\sqrt{1-x^2}}Q(x)\right)
\end{aligned}$$

Thus, clearly if $deg(Q(x)) \leq d-2 \implies deg(Q'(x)) \leq d-1$ because the highest order terms $P(x)$ and also $\frac{x}{\sqrt{1-x^2}}Q(x)$. Both of which correspond to a degree $\leq d-1$.

We can also prove the condition (2) by taking different cases that let d be even then proving that $P'(x)$ is even. Let d be even, then parity of $P(x)$ is $d-1 \bmod 2 \equiv 1$ i.e. $P(x)$ is an odd function $\implies P(-x) = -P(x)$ and similarly the parity of $Q(x)$ is $d-2 \bmod 2 \equiv 0$ i.e. $Q(x)$ is an even function $\implies Q(-x) = Q(x)$. Thus, we need to show that $P'(-x) = P'(x)$ in order to prove that $P'(x)$ has parity $d \bmod 2 \equiv 0$ i.e. $P'(x)$ is an even function.

$$P'(-x) = e^{\iota\phi}(-xP(-x) - (1-(-x)^2)Q(-x))$$

Now substituting the results $P(-x) = -P(x)$ and $Q(-x) = Q(x)$, we get,

$$P'(-x) = e^{\iota\phi}(xP(x) - (1-x^2)Q(x)) = P'(x)$$

Thus, $P'(x)$ is an even function i.e. parity 0 which corresponds to parity of $d \bmod 2 \equiv 0$. Similarly we can show that $Q'(-x) = -Q'(x)$ i.e. it corresponds to a parity $d-1 \bmod 2 \equiv 1$ an odd function. Thus,

$$\begin{aligned} Q'(x) &= e^{\iota\phi}\left(P(x) + \frac{x}{\sqrt{1-x^2}}Q(x)\right) \\ Q'(-x) &= e^{\iota\phi}\left(P(-x) + \frac{-x}{\sqrt{1-(-x)^2}}Q(-x)\right) \end{aligned}$$

Now substituting the facts that $P(-x) = -P(x)$ and $Q(-x) = Q(x)$, we get

$$\begin{aligned} Q'(-x) &= e^{\iota\phi}\left(P(-x) + \frac{-x}{\sqrt{1-(-x)^2}}Q(-x)\right) \\ &= e^{\iota\phi}\left(-P(x) + \frac{-x}{\sqrt{1-x^2}}Q(x)\right) \\ &= -e^{\iota\phi}\left(P(x) + \frac{x}{\sqrt{1-x^2}}Q(x)\right) \\ &= -Q'(x) \end{aligned}$$

Thus, $Q'(-x) = -Q'(x)$ i.e. an odd parity which corresponds to parity $d-1 \bmod 2 \equiv 1$, an odd function.

Now, let d be odd, then parity of $P(x)$ is $d-1 \bmod 2 \equiv 0$ i.e. $P(x)$ is an even function $\implies P(-x) = P(x)$ and similarly the parity of $Q(x)$ is $d-2 \bmod 2 \equiv 1$ i.e. $Q(x)$ is an odd function $\implies Q(-x) = -Q(x)$. Thus, we need to show that $P'(-x) = -P'(x)$ in order to prove that $P'(x)$ has parity $d \bmod 2 \equiv 1$ i.e. $P'(x)$ is an odd function.

$$\begin{aligned} P'(x) &= e^{\iota\phi}(xP(x) - (1-x^2)Q(x)) \\ P'(-x) &= e^{\iota\phi}(-xP(-x) - (1-(-x)^2)Q(-x)) \end{aligned}$$

Now substituting the results $P(-x) = P(x)$ and $Q(-x) = -Q(x)$, we get,

$$P'(-x) = e^{\iota\phi}(-xP(x) + (1-x^2)Q(x)) = -P'(x)$$

Thus, $P'(x)$ is an odd function i.e. parity 1 which corresponds to parity of $d \mod 2 \equiv 1$. Similarly we can show that $Q'(-x) = Q'(x)$ i.e. it corresponds to a parity $d-1 \mod 2 \equiv 0$ an even function. Thus,

$$\begin{aligned} Q'(x) &= e^{\iota\phi}\left(P(x) + \frac{x}{\sqrt{1-x^2}}Q(x)\right) \\ &= e^{\iota\phi}\left(P(-x) + \frac{-x}{\sqrt{1-(-x)^2}}Q(-x)\right) \end{aligned}$$

Now substituting the facts that $P(-x) = P(x)$ and $Q(-x) = -Q(x)$, we get

$$\begin{aligned} Q'(-x) &= e^{\iota\phi}\left(P(-x) + \frac{-x}{\sqrt{1-(-x)^2}}Q(-x)\right) \\ &= e^{\iota\phi}\left(P(x) - \frac{-x}{\sqrt{1-x^2}}Q(x)\right) \\ &= e^{\iota\phi}\left(P(x) + \frac{x}{\sqrt{1-x^2}}Q(x)\right) \\ &= Q'(x) \end{aligned}$$

Thus, $Q'(-x) = Q'(x)$ i.e. an even parity which corresponds to parity $d-1 \mod 2 \equiv 0$, an even function.

Thus, we showed that if d is even which implies that $P(x)$ and $Q(x)$ correspond to odd and even functions respectively, then, $P'(x)$ and $Q'(x)$ corresponds to even and odd functions respectively. Similarly if d is odd which implies that $P(x)$ and $Q(x)$ correspond to even and odd functions respectively, then, $P'(x)$ and $Q'(x)$ corresponds to even and odd functions respectively. Thus, verifying condition (2).

Because it is an if and only if condition we now prove the other way around. When $d = 0$ the only possibility is $P(x) = e^{\iota\phi_0}$ and $Q = 0$, which satisfies the all the three conditions.

For $d > 0$, when d is even we may still have $deg(P) = 0$ i.e., $P(x) = e^{\iota\phi_0}$. In this case, note that

$$O^{-1}(x) = O^{\dagger}(x) = \begin{pmatrix} x & \sqrt{1-x^2} \\ -\sqrt{1-x^2} & x \end{pmatrix} = e^{-\iota\frac{\pi}{2}Z}O(x)e^{\iota\frac{\pi}{2}Z}$$

we may set $\phi_j = (-1)^j\frac{\pi}{2}, j = 1, \ldots, d$, and

$$e^{\iota\phi_0 Z}\prod_{j=1}^{d}[O(x)e^{\iota\phi_j Z}] = e^{\iota\phi_0 Z}(O(x)O^{\dagger}(x))^{\frac{d}{2}} = e^{\iota\phi_0 Z}$$

Thus, the statement holds.

Now given P, Q satisfying the conditions $(1)-(3)$, with $deg(P) = l > 0$ and $l \equiv d(\text{mod } 2)$ (meaning that $l - d = 2k$, thus $l - d$ is even, which means if l is odd then d is odd and l is even than d is even). Then $deg(|P(x)|^2) = 2l > 0$, and according to the condition (3) we must have $deg(Q) = l - 1$. This is so that in order for their sum to be 1 they powers must be the same so that they cancel out. Thus, degree of $(1-x^2)|Q(x)|^2$ will be same as degree of $x^2|Q(x)|^2$. The degree of $|P(s)|^2$ is $2l$. Hence, the $deg(Q(x)) = l-1$ so that $deg(x^2|Q(x)|^2) = 2l$ which can cancel out the terms in $|P(x)|^2$ in order to sum to 1. Let P, Q be expanded as

$$P(x) = \sum_{k=0}^{l} \alpha_k x^k, \quad Q(x) = \sum_{k=0}^{l-1} \beta_k x^k$$

then the leading term of $|P(x)|^2 + (1-x^2)|Q(x)|^2 = 1$ is

$$|\alpha_l|^2 x^{2l} - x^2|\beta_{l-1}|^2 x^{2l-2} = (|\alpha_l|^2 - |\beta_{l-1}|^2)x^{2l} = 0$$

which implies $|\alpha_l| = |\beta_{l-1}|$.

For any $\phi \in \mathbb{R}$, we have

$$\begin{aligned}
&\begin{pmatrix} P(x) & -Q(x)\sqrt{1-x^2} \\ Q^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix} e^{-\iota\phi Z} O^\dagger(x) \\
&= \begin{pmatrix} P(x) & -Q(x)\sqrt{1-x^2} \\ Q^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix} \begin{pmatrix} e^{-\iota\phi} & 0 \\ 0 & e^{\iota\phi} \end{pmatrix} \begin{pmatrix} x & \sqrt{1-x^2} \\ -\sqrt{1-x^2} & x \end{pmatrix} \\
&= \begin{pmatrix} e^{-\iota\phi}xP(x) + (1-x^2)Q(x)e^{\iota\phi} & -\sqrt{1-x^2}(-e^{-\iota\phi}P(x) + xQ(x)e^{\iota\phi}) \\ \sqrt{1-x^2}(-e^{-\iota\phi}P^*(x) + xQ^*(x)e^{-\iota\phi}) & e^{\iota\phi}xP^*(x) + (1-x^2)Q^*(x)e^{-\iota\phi} \end{pmatrix} \\
&= \begin{pmatrix} \tilde{P}(x) & -\tilde{Q}(x)\sqrt{1-x^2} \\ \tilde{Q}^*(x)\sqrt{1-x^2} & \tilde{P}^*(x) \end{pmatrix}
\end{aligned}$$

It may appear that $deg(\tilde{P}) = l + 1$. However by properly choosing ϕ we may obtain $deg(\tilde{P}) = l - 1$. Let $e^{2\iota\phi} = \alpha_l/\beta_{l-1}$. Then the coefficient of the x^{l+1} term in $\tilde{P}$ is

$$e^{-\iota\phi}\alpha_l - e^{\iota\phi}\beta_{l-1} = \frac{\alpha_l + e^{2\iota\phi}\beta_{l-1}}{e^{\iota\phi}} = 0$$

Similarly, the coefficient of the x^l term in $\tilde{Q}$ is

$$-e^{\iota\phi}\alpha_l + e^{\iota\phi}\beta_{l-1} = 0$$

The coefficient of the x^l term in $\tilde{P}$, and the coefficient of the x^{l-1} term in $\tilde{Q}$ are both 0 by the parity condition. So we have

1. $deg(\tilde{P}) \leq l-1 \leq d-1$, $deg(Q) \leq l-2 \leq d-2$.
2. $\tilde{P}$ has parity $d-1$ mod 2 and $\tilde{Q}$ has parity $d-2$ mod 2, and
3. $|\tilde{P}(x)|^2 + (1-x^2)|\tilde{Q}(x)|^2 = 1$, $\forall x \in [-1,1]$.

Here the condition (3) is automatically satisfies due to unitary. The induction follows until $l=0$, and applying the argument to represent the remaining constant phase factor if needed. □

Remark. **(W convention of QSP)** It is stated slightly differently as

$$U_{\Phi W}(x) = e^{\iota\phi_0^W Z}\prod_{j=1}^{d}[W(x)e^{\iota\phi_j^W Z}] = \begin{pmatrix} P(x) & \iota Q(x)\sqrt{1-x^2} \\ \iota Q^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix}$$

where

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

This will be referred to as the W-convention. Correspondingly the previous equation will be referred to as the O-convention. The two conventions can be easily converted into one another, due to the relation

$$\begin{aligned} W(x) &= e^{-\iota\frac{\pi}{4}Z}O(x)e^{\iota\frac{\pi}{4}Z} \\ &= \begin{bmatrix} e^{-\iota\pi/4} & 0 \\ 0 & e^{\iota\pi/4} \end{bmatrix}\begin{bmatrix} x & -\sqrt{1-x^2} \\ \sqrt{1-x^2} & x \end{bmatrix}\begin{bmatrix} e^{\iota\pi/4} & 0 \\ 0 & e^{-\iota\pi/4} \end{bmatrix} \\ &= \begin{bmatrix} e^{-\iota\pi/4}x & -e^{-\iota\pi/4}\sqrt{1-x^2} \\ e^{\iota\pi/4}\sqrt{1-x^2} & e^{\iota\pi/4}x \end{bmatrix}\begin{bmatrix} e^{\iota\pi/4} & 0 \\ 0 & e^{-\iota\pi/4} \end{bmatrix} \\ &= \begin{bmatrix} x & -e^{-\iota\pi/2}\sqrt{1-x^2} \\ e^{\iota\pi/2}\sqrt{1-x^2} & x \end{bmatrix} \\ &= \begin{bmatrix} x & \iota\sqrt{1-x^2} \\ \iota\sqrt{1-x^2} & x \end{bmatrix} = W(x) \end{aligned}$$

Correspondingly the relation between the phase angles using the O and W represen-

tation are related according to

$$\begin{aligned}
U_{\Phi^W}(x) &= e^{\iota\phi_0^W}\prod_{j=1}^{d}[W(x)e^{\iota\phi_j^W Z}] \\
&= e^{\iota\phi_0^W Z}\prod_{j=1}^{d}[e^{-\iota\frac{\pi}{4}Z}O(x)e^{\iota\frac{\pi}{4}Z}e^{\iota\phi_j^W Z}] \\
&= e^{\iota\phi_0^W Z}e^{-\iota\frac{\pi}{4}Z}O(x)\dots O(x)e^{\iota\frac{\pi}{4}Z}e^{\iota\phi_j^W Z}e^{-\iota\frac{\pi}{4}Z}O(x)\dots O(x)e^{-\iota\frac{\pi}{4}Z}e^{\iota\phi_d^W Z} \\
&= e^{\iota(\phi_0^W-\frac{\pi}{4})Z}O(x)\dots O(x)e^{\iota(\frac{\pi}{4}+\phi_j^W-\frac{\pi}{4})Z}O(x)\dots O(x)e^{\iota(\phi_d^W-\frac{\pi}{4})Z} \\
&= e^{\iota(\phi_0^W-\frac{\pi}{4})Z}O(x)\dots O(x)e^{\iota\phi_j^W Z}O(x)\dots O(x)e^{\iota(\phi_d^W-\frac{\pi}{4})Z}
\end{aligned}$$

which simplifies to,

$$\phi_j = \begin{cases} \phi_0^W - \frac{\pi}{4}, & \text{if } j = 0 \\ \phi_j^W, & \text{if } j = 1,\dots,d-1 \\ \phi_d^W + \frac{\pi}{4}, & \text{if } j = d \end{cases}$$

On the other hand, note that for any $\theta \in \mathbb{R}, U_\Phi(x)$ and $e^{\iota\theta Z}U_\Phi(x)e^{-\iota\theta Z}$ both block encodes $P(x)$. This can be seen as follows:

$$\begin{aligned}
U_\Phi(x) &= \begin{bmatrix} P(x) & -Q(x)\sqrt{1-x^2} \\ Q^*(x)\sqrt{1-x^2} & P^*(x) \end{bmatrix} \\
e^{\iota\theta Z}U_\Phi(x)e^{-\iota\theta Z} &= \begin{bmatrix} e^{\iota\theta} & 0 \\ 0 & e^{-\iota\theta} \end{bmatrix}\begin{bmatrix} P(x) & -Q(x)\sqrt{1-x^2} \\ Q^*(x)\sqrt{1-x^2} & P^*(x) \end{bmatrix}\begin{bmatrix} e^{-\iota\theta} & 0 \\ 0 & e^{\iota\theta} \end{bmatrix} \\
&= \begin{bmatrix} e^{\iota\theta}P(x) & -e^{\iota\theta}Q(x)\sqrt{1-x^2} \\ e^{-\iota\theta}Q^*(x)\sqrt{1-x^2} & e^{-\iota\theta}P^*(x) \end{bmatrix}\begin{bmatrix} e^{-\iota\theta} & 0 \\ 0 & e^{\iota\theta} \end{bmatrix} \\
&= \begin{bmatrix} P(x) & -e^{2\iota\theta}Q(x)\sqrt{1-x^2} \\ e^{-2\iota\theta}Q^*(x)\sqrt{1-x^2} & P^*(x) \end{bmatrix}
\end{aligned}$$

Thus, both block encode $P(x)$. Therefore without loss of generality we may as well take

$$\Phi = \Phi^W$$

In many applications, we are only interested in $P \in \mathbb{C}[x]$, and $Q \in \mathbb{C}[x]$ is not provided a priori. The theorem in [1] states that under certain conditions P, the polynomial Q can always be constructed. We omit the details here.

6.2 QSP for real polynomials

Note that the normalization condition (3) in theorem imposes very strong constraints on the coefficients of $P, Q \in \mathbb{C}[x]$. If we are only interested in QSP for real polynomials, the condition can be significantly relaxed.

Theorem 6.2.1. ***(Quantum signal processing for real polynomials)*** *Given a real polynomial $P_{Re}(x) \in \mathbb{R}[x]$, and $deg(P_{Re}) = d > 0$ satisfying*

1. *P_{Re} has parity d mod 2.*
2. *$|P_{Re}| \leq 1 \forall x \in [-1, 1]$*

then there exists polynomials $P(x), Q(x) \in \mathbb{C}[x]$ with $Re(P) = P_{Re}$ and a set of phase factors $\Phi = (\phi_0, \ldots, \phi_d) \in \mathbb{R}^{d+1}$ such that the QSP representation holds.

Compared to the theorem 6.1.1, the conditions in Theorem 6.2.1 is much easier to satisfy: given any polynomial $f(x) \in \mathbb{R}[x]$ satisfy condition (1) on parity, we can always scale f to satisfy the condition (2) on its magnitude. We can now summarize the result of QET with real polynomials as follows.

Corollary 6.2.2 (Quantum eigenvalue transformation with real polynomials)**.** *. Let $A \in \mathbb{C}^{N \times N}$ be encoded by its $(1, m)$-block-encoding U_A. Given a polynomial $P_{Re}(x) \in \mathbb{R}[x]$ of degree d satisfying the conditions in Theorem 6.2.1, we can find a sequence of phase factors $\Phi \in \mathbb{R}^{d+1}$, so that the circuit in figure 5.5 denoted by U_Φ implements a $(1, m+1)$-block-encoding of $P_{Re}(A)$. U_Φ uses $U_A, U_A^\dagger$, m-qubit controlled NOT, and single qubit rotation gates for $O(d)$ times.*

Remark. (Relation between QSP representation and QET circuit). Although $O(x) = U_A(x)Z$, we do not actually need to implement Z separately in QET. Note that $\iota Z = e^{\iota \frac{\pi}{2} Z}$,i.e. $Ze^{\iota\phi Z} = (-\iota)e^{\iota(\frac{\pi}{2}+\phi)Z}$. we obtain

$$\begin{aligned} U_\Phi(x) &= e^{\iota\phi_0 Z} \prod_{j=1}^{d} [O(x)e^{\iota\phi_j Z}] \\ &= e^{\iota\phi_0 Z} \prod_{j=1}^{d} [U_A(x) Z e^{\iota\phi_j Z}] \\ &= e^{\iota\phi_0 Z} \prod_{j=1}^{d} [U_A(x)(-\iota) e^{\iota(\frac{\pi}{2}+\phi)Z}] \\ &= (-\iota)^d e^{\iota\tilde{\phi}_0 Z} \prod_{j=1}^{d} [U_A(x) e^{\iota\tilde{\phi}_j Z}] \end{aligned}$$

where $\tilde{\phi}_0 = \phi_0, \tilde{\phi}_j = \phi_j + \pi/2; j = 1, \ldots, d$, which implies,

$$\tilde{\phi}_j = \begin{cases} \phi_0, & j = 0 \\ \phi_j + \frac{\pi}{2}, & j = 1, \ldots d \end{cases}$$

For the purpose of block encoding $P(x)$, recall that $e^{\iota\theta Z} U_\Phi(x) e^{-\iota\theta Z}$, another equivalent, and more symmetric choice with $\theta = \pi/4$ is

$$\tilde{\phi}_j = \begin{cases} \phi_0 + \frac{\pi}{4}, & j = 0 \\ \phi_j + \frac{\pi}{2}, & j = 1, \ldots, d-1 \\ \phi_d + \frac{\pi}{4}, & j = d \end{cases}$$

When the phase factors are given in the W-convention, since we can perform a similarity transformation and take $\Phi = \Phi^W$, we can directly convert Φ^W to $\tilde{\Phi}$ according to this equation, which is used in the QET circuit in figure!5.4.

Example 6.2.1. (QSP for Chebyshev polynomials revisited). In order to block encode the Chebyshev polynomial, we have $\phi_j = 0, j = 0, \ldots, d$. This gives $\tilde{\phi}_0 = 0, \tilde{\phi}_j = \frac{\pi}{2}, j = 1, \ldots, d$, and

$$U_\Phi(x) = [O(x)]^d = \begin{bmatrix} T_d(x) & -\sqrt{1-x^2} U_{d-1}(x) \\ \sqrt{1-x^2} U_{d-1}(x) & T_d(x) \end{bmatrix} = (-\iota)^d \prod_{j=1}^{d} [U_A(x) e^{\iota \frac{\pi}{2} Z}]$$

According to the equation, an equivalent symmetric choice for block encoding $T_d(x)$ is

$$\tilde{\phi}_j = \begin{cases} \frac{\pi}{4}, j = 0 \\ \frac{\pi}{2}, j = 1, \ldots, d-1 \\ \frac{\pi}{4}, j = d \end{cases}$$

6.3 Optimization based method for finding phase factors

QSP for real polynomials is the most useful version for many problems in scientific computation. Let us not summarize the problem of finding phase factors following the W-convention and identify $\phi = \Phi^W$. Given a target polynomial $f = P_{Re} \in \mathbb{R}[x]$ satisfying (1)$deg(f) = d$, (2) the parity of f is d mod 2, (3) $\|f\|_\infty = \max_{x \in [-1,1]} |f(x)| < 1$, we would like to find phase factors $\Phi = (\phi_0, \ldots, \phi_d) \in [-\pi, \pi)^{d+1}$ so that

$$f(x) = g(x, \Phi) = Re[U(x, \Phi)_{11}], \quad x \in [-1, 1]$$

with

$$U(x,\Phi) = e^{\iota\phi_0 Z} W(x) e^{\iota\phi_1 Z} W(x) \dots e^{\iota\phi_{d-1} Z} W(x) e^{\iota\phi_d Z}$$

Theorem 6.2.1 shows the existence of the phase factors. Due to the parity constraint, the number of degrees of freedom in the target polynomial $f(x)$ is $\tilde{d} = \lceil \frac{d+1}{2} \rceil$. This can be verified by counting the number of free variables in $f(x)$, if $f(x)$ is of degree d and parity $d \bmod 2$. Hence $f(x)$ is entirely determined by its values on $\tilde{d}$ distinct points. Since if the values at $\tilde{d}$ points is known then we can formulate a system of linear equations whose solutions will be the coefficients of the function $f(x)$. Through the paper, we choose these points to be $x_k = \cos(\frac{2k-1}{4\tilde{d}}\pi), k = 1, \dots, \tilde{d}$, i.e., positive nodes of the Chebyshev polynomial $T_{2\tilde{d}}(x)$. The QSP problem can be equivalently solved via the following optimization problem

$$\Phi^* = \arg \min_{\Phi \in [-\pi,\pi)^{d+1}} F(\Phi), \quad F(\Phi) = \frac{1}{\tilde{d}} \sum_{k=1}^{\tilde{d}} |g(x_k,\Phi) - f(x_k)|^2$$

i.e., any solution Φ^* achieves the global minimum of the cost function with $F(\Phi^*) = 0$, and vice versa.

However, note that the number of variables (for finding the phase $\Phi = (\phi - 0, \dots, \phi_d) \in \mathbb{R}^{d+1}$) $d+1$ is larger than the number of equations $\tilde{d} = \lceil \frac{d+1}{2} \rceil$ and there should be infinite number of global minima. It can be shows that the existence of symmetric phase factors

$$\Phi = (\phi_0, \phi_1, \phi_2, \dots, \phi_2, \phi_1, \phi_0) \in [-\pi, \pi)^{d+1}$$

Thus, now we have equal number of constraints and variables. Then the optimization problem is changed to

$$\Phi^* = \arg \min_{\Phi \in [-\pi,\pi)^{d+1}} F(\Phi), \quad F(\Phi) = \frac{1}{\tilde{d}} \sum_{k=1}^{\tilde{d}} |g(x_k,\Phi) - f(x_k)|^2$$

This corresponds to choosing complementary polynomial $Q(x) \in \mathbb{R}[x]$. With the symmetric constraint taken into account, the number of variables matches the number of constraints.

Unfortunately, the energy landscape of the cost function $F(\Phi)$ is very complex, and has numerous global as well as local minima. Starting from a random initial guess, an optimization algorithm can easily be trapped at a local minima already when d is small. It is therefore surprising that starting from a special symmetric initial guess

$$\Phi^0 = (\pi/4, 0, 0, \dots, 0, 0, \pi/4)$$

at least one global minimum can be robustly identified using standard unconstrained optimization algorithms even when d is as large as 10000 using standard double precision arithmetic operations, and the optimization method is observed to be free from being trapped by any local minima. Direct calculation shows that $g(x, \Phi^0) = 0$,

$$\begin{aligned} g(x,\Phi^0) &= Re[U(x,\Phi^0)_{11}] \quad \forall x \in [-1,1] \\ &= Re[(e^{\iota\phi_0 Z}W(x)e^{\iota\phi_1 Z}W(x)\dots e^{\iota\phi_{d-1}Z}W(x)e^{\iota\phi_d Z})_{11}] \\ &= Re[(e^{\iota\frac{\pi}{4}Z}W(x)W(x)\dots W(x)e^{\iota\frac{\pi}{4}Z})_{11}] \\ &= Re[(e^{\iota\frac{\pi}{4}Z}W(x)^d e^{\iota\frac{\pi}{4}Z})_{11}] \\ &= Re[(e^{\iota\frac{\pi}{4}Z}e^{\iota d\arccos(x)X}e^{\iota\frac{\pi}{4}Z})_{11}] \\ &= Re(e^{\iota\frac{\pi}{2}}T_d(x)) \\ &= 0 \end{aligned}$$

and therefore Φ^0 does not contain apriori information of the target polynomial $f(x)$. This optimization based method is implemented in QSPPACK.

Remark. (Other methods for treating QSP with real polynomials). The proof of [1] also gives a constructive algorithm for solving the QSP problem for real polynomials so that the resulting $P(x) = P_{Re}(x) + \iota P_{Im}(x)$ and $Q(x)$ satisfy the requirement in theorem 6.1.1. Then the phase factors can be constructed following the recursion relation shown in the proof of theorem 6.1.1. We will not describe the details of the procedure here. It is worth nothing that the method is not numerically stable. This is made more precise that these algorithms require $O(d\log(d/\epsilon))$ bits of precision, where d is the degree of $f(x)$ and ϵ is the largest accuracy. It is worth mentioning that the extended precision needed in these algorithms is not an artifact of the proof techique. For instance, for $d \approx 500$, the number of bits needed to present each floating point numbers can be as large as $1000 - 2000$. In particular, such a task cannot be reliably performed using standard double precision arithmetic operations which only has 64 bits.

6.4 A typical workflow preparing the circuit of QET

Let us now use $f(x) = \cos(xt)$ as in the Hamiltonian simulation to demonstrate a typical workflow of QSP. This function should be an even or odd function to satisfy the parity constrain (1) in theorem 6.2.1.

1. Expand $f(x), x \in [-1, 1]$ using a polynomial expansion (in this case, the Jacobi-Anger expansion), and truncate it to some finite order.
2. Scale the truncated polynomial by a suitable constant so that the resulting polynomial $P_{Re}(x)$ satisfies the max-norm constraint 92) in Theorem 6.2.1.
3. Use the optimization based method to find phase factors $\Phi^W = \Phi$, and convert the result to $\tilde{\Phi}$ according to the relation. $\tilde{\Phi}$ can be directly used in the QET circuit in figure 5.3 and '5.4.

6.5 Dominant resource cost - gates and qubits

A QSP circuit that implements a degree d polynomials 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 polynomials, either using high-precision arithmetic with $\approx d \log(d)$ bits of precision (or more - though this can be mitigated using heuristic techniques) or in some regimes using more efficient optimization-based algorithms. Although these procedures are efficient in theory, in practice it may still be non-trivial to find the angles. Nevertheless, researches reportedly computed angle sequences corresponding to various degree $\mathcal{O}(10^4)$ polynomials.

6.6 Caveats

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

- $Deg(P) \leq d$, $Deg(Q) \leq d - 1$
- $Parity(P) = Parity(d)$, $Parity(Q) = Parity(d - 1)$
- $\forall x \in [-1, 1] : |P(x)|^2 + (1 - x^2)|Q(x)|^2 = 1$ (for the matrix 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 elements $\langle +| U_{QSP}(\Phi) |+\rangle$ rather than in $\langle 0| U_{QSP}(\Phi) |0\rangle$, where $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. This matrix element evaluates to

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

Given a real target polynomial $f(x)$ with parity equal to $parity(d)$, we can guarantee that the matrix element evalulates to $f(x)$ by choosing $Re[P(x)] = f(x)$ and $Re[Q(x)] = 0$. The third condition above then reduces to $1 - f(x)^2 = Im[P(x)]^2 + (1 - x^2)|Im[Q(x)]|^2$. There exist choice for $Im[P(x)]$ and $Im[Q(x)]$ 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

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

There are sevel related conventions considered in the literature for the explicit form of the signal qubit operators used in QSP. ONe common form that links closely to qubitization and QSVT is the reflection convention, which replced $W(x)$ by the reflection

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

and adjusts the parameters $\{\phi_j\}$ accordingly.

6.7 Examples

6.7.1 Functions of Hermitian/normal matrices

Functions of Hermitian/normal matrices, in conjunction with qubitization, including for Hamiltonian simulation.

Example 6.7.1. (Hamiltonian simulation)

Important Note

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 $U_{QSP}(\Phi)$ given above, the achievable polynomial pairs $P(x), Q(x) \in \mathbb{C}$ can be characterized by the following three conditions:

- $Deg(P) \leq d, \quad Deg(Q) \leq d-1$
- $Parity(P) = Parity(d), \quad Parity(Q) = Parity(d-1)$
- $\forall xin[-1,1] : |P(x)|^2 + (1-x^2)|Q(x)|^2 = 1$ (required to be Unitary)

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

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

Given a real target polynomial $f(x)$ with parity equal to $Parity(d)$, we can guarantee that the matrix element evaluates to $f(x)$ by choosing $Re[P(x)] = f(x)$ and $Re[Q(x)] = 0$. The third condition above then reduces to $1 - f(x)^2 = |Im[P(x)]|^2 + (q - x^2)|Im[Q(x)]|^2$. There exist choices for $Im[P(x)]$ and $Im[Q(x)]$ 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.

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

There are several related conventions considered in the literature for the explicit form of the single qubit operators used in QSP. One common form that links closely to qubitization and QSVT is the reflection conventions, which replaces $W(x)$ by the reflection

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

and adjusts the parameters $\{\phi_j\}$ accordingly.

Some example uses cases include:

- Functions of Hermitian/normal matrices, in conjunction with qubitization, including for Hamiltonian simulation.
- Functions of general matrices via quantum singular value transformation (QSVT).
- QSP applied to beyond-Heisenberg limit calibration of two-qubit gates in a superconducting system.

Chapter 7

Quantum Singular Value Transformation

In previous chapters, we have found that using qubitization, we can effectively block encode the Chebyshev matrix polynomial $T_k(A)$ for a Hermitian matrix A. Combined with LCU, we can construct a block encoding of any matrix polynomial of A. The process is greatly simplified using QSP and QET, which allows the implementation of a general class of matrix functions for Hermitian matrices.

In this section, we generalize the results of qubitization and QET to general non-Hermitian matrix. This is called the quantum singular value transformation (QSVT). Throughout the chapter we assume $A \in \mathbb{C}^{N\times N}$ is a square matrix. QSVT is applicable to non-square matrices as well, and we will omit the discussions here.

7.1 Generalized Matrix functions

For a square matrix $A \in \mathbb{C}^{N\times N}$, wher for simplicity we assume $N = 2^n$ for some positive integer n, the singular value decomposition (SVD) of the normalized matrix A can be written as

$$A = W\Sigma V^\dagger$$

or equivalently

$$A\left|v_i\right\rangle = \sigma_i \left|w_i\right\rangle, \quad A^\dagger \left|w_i\right\rangle = \sigma_i \left|v_i\right\rangle, \quad i \in [N]$$

We may apply a function $f(\cdot)$ on its singular values and define the generalized matrix function as below.

Definition 7.1.1. (Generalized matrix function) Given $A \in \mathbb{C}^{N\times N}$ with singular value decomposition as given above, and let $f : \mathbb{R} \to \mathbb{C}$ be a scalar function such

that $f(\sigma_i)$ is defined for all $i \in [N]$. The generalized matrix function is defined as

$$f^o(A)$$

Bibliography

[1] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, pages 193–204, 2019.