

ALMA MATER STUDIORUM · UNIVERSITY OF BOLOGNA

SCHOOL OF SCIENCE
Degree Programme in Mathematics

Block encoding techniques:
an explicit quantum circuit for the Heisenberg
Hamiltonian

Master's Thesis in Mathematics

Supervisor:
Prof. Giacomo De Palma

Presented by:
Filippo Della Chiara

Co-supervisor:
Prof. Valeria Simoncini

Academic Year 2024–2025

“Il tempo vola e porta con sé via tutto”

Contents

1	Introduction	1
2	Preliminaries on quantum computing	7
2.1	Quantum Bits	7
2.2	Quantum measurement	14
2.3	Quantum Circuit	17
3	Quantum singular value transformation	23
3.1	Quantum signal processing	24
3.2	Amplitude Amplification	28
3.3	Quantum eigenvalue transform	31
4	Block encodings	37
4.1	General notion	37
4.2	Fast Approximate Quantum Circuits for Block-Encodings . . .	45
4.3	Sparse matrices	51
5	Efficient Block Encoding of Spin Hamiltonians	61
5.1	Fast One-Qubit Control Select LCU	62
5.2	Preparation of Dicke states	70
5.3	Applications on spin models	80
5.4	Compression of P_R for the Heisenberg Model	86
6	Conclusion	93

Bibliography

95

List of Figures

2.1 Representation of a single-qubit pure state $ \psi\rangle$ on the Bloch sphere.	10
3.1 Illustration of amplitude amplification on the Bloch sphere: starting from $ B_0\rangle$, controlled rotations about $ B_0\rangle$ (blue) and $ A_0\rangle$ (black) drive the state toward the target $ A_0\rangle$ (north pole).	29
4.1 A complete quantum circuit for the block encoding of a 8×8 banded circulant matrix.	60
5.1 LCU block encoding	63
5.2 Circuits representing FOQCS-LCU.	68
5.3 Circuit used to implement P_R for the Heisenberg model.	83
5.4 Decomposition of the gates in fig. 5.3.	88
5.5 Commutations of the controlled gates of fig. 5.4.	89
5.6 Compression of controlled Dicke gates.	89
5.7 Compact implementation of P_R for the Heisenberg model	90
5.8 CNOT count from FOQCS-LCU, LCU, and FABLE for the Heisenberg model, with FABLE evaluated at precisions $\epsilon = 10^{-3}$ and $\epsilon = 10^{-6}$.	92

Abstract

Quantum computing leverages quantum mechanics to solve problems that are intractable for classical computers. A central framework in this domain is Quantum Singular Value Transformation (QSVT), which enables efficient manipulation of matrix operations and underlies many quantum algorithms, including those for solving linear systems and simulating quantum dynamics.

Block encoding is a key technique that embeds non-unitary matrices into unitary operators, making them amenable to the QSVT. Among block encoding methods, the Linear Combination of Unitaries (LCU) technique is widely used, but its practical utility is limited by high gate overhead—particularly from multi-controlled operations.

This thesis introduces a new formulation, FOQCS-LCU, which reduces both practical and asymptotic circuit complexity. By exploiting the structure of physically motivated Hamiltonians, we also develop efficient routines for preparing Dicke states, which are superpositions over basis states with fixed Hamming weight.

We demonstrate our method by constructing explicit block encoding circuits for the Heisenberg Hamiltonian, achieving an order-of-magnitude reduction in CNOT gate count compared to standard LCU approaches. Detailed gate counts and numerical benchmarks confirm the efficiency of our technique. This work advances the feasibility of block encoding as a subroutine for large-scale quantum algorithms and supports more efficient implementations on fault-tolerant quantum devices.

Chapter 1

Introduction

Quantum computing [1] is an emerging field at the intersection of computer science, physics, and engineering. It leverages the principles of quantum mechanics to address computational problems that are currently intractable for even the most powerful classical supercomputers. Unlike classical computers—which rely on binary logic and conventional hardware—quantum computers harness phenomena such as superposition and entanglement to process information in fundamentally new ways. This enables quantum algorithms to offer significant speedups for certain tasks. For instance, problems that would take a classical computer thousands of years to solve may be tackled by a quantum device in minutes or hours. While classical computers are expected to remain the standard for most computational tasks, quantum advantage becomes evident in solving highly complex problems. These include simulating quantum systems such as molecules and materials, optimizing large-scale systems, or modeling interactions in particle physics [2]. Such problems are challenging for classical machines due to the exponential growth in computational resources required as the system size increases.

Quantum algorithms use the formalism of *quantum circuits* to describe how quantum information is processed. A quantum circuit is formally defined as a sequence of quantum operations (gates) applied to qubits. While often represented using graphical diagrams, the circuit itself is the underlying

ing sequence of unitary transformations and measurements. In this model, qubits are represented by *wires*, and quantum operations are represented by *gates* placed along these wires. As quantum information flows through the circuit, gates transform the state of the qubits at specific positions, implementing a sequence of quantum operations. Each quantum gate corresponds to a unitary matrix, ensuring that the transformation preserves the norm of the quantum state. This unitarity condition guarantees that the quantum state remains normalized—i.e., with norm equal to 1—throughout the computation.

We provide a detailed overview of the powerful framework known as Quantum Singular Value Transformation (QSVT) [3, 4]. This framework has become fundamental in the field, as it unifies many important quantum algorithms under a single formalism. QSVT enables the application of polynomial transformations to the singular values of a target matrix, which must be embedded within a larger unitary operator to be processed on a quantum computer.

QSVT provides a versatile and unified approach to realizing a wide range of quantum algorithms. One of its most important applications is in solving systems of linear equations, a fundamental computational problem. QSVT-based algorithms generalize and improve upon the Harrow-Hassidim-Lloyd (HHL) algorithm ([5]) by offering greater flexibility and better precision in encoding and manipulating matrix operations. Another major application is in quantum simulation [6, 7], particularly for simulating the time evolution of quantum systems. In quantum mechanics, the time evolution of a closed quantum system is governed by the Schrödinger equation, where the Hamiltonian operator H determines the system's dynamics. Simulating time evolution amounts to implementing the unitary operator e^{-iHt} , which describes how a quantum state evolves after time t under the influence of the Hamiltonian H . This task is computationally demanding for classical computers, especially when H describes a many-body or interacting system. QSVT enables efficient approximation of such exponential operators, opening

the door to practical quantum simulations of physical systems [8, 9], including molecules, materials, and strongly correlated quantum matter. These capabilities make QSVT a cornerstone of modern quantum algorithm design, providing a practical tool for advancing quantum computing applications.

This thesis focuses on block encoding, which is a technique that enables the efficient embedding of matrices into higher-dimensional unitary operators, which is the requirement for applying Quantum Singular Value Transformation (QSVT). Block encoding allows a non-unitary matrix to be represented as a submatrix—typically in the top-left corner—of a unitary operator that can be implemented as a quantum circuit. Block encoding not only facilitates the use of QSVT but is foundational to its operation: once a matrix is block-encoded, QSVT can apply designed polynomial transformations to its singular values. However, constructing a block encoding is not merely a theoretical step—it requires an explicit quantum circuit implementing the larger unitary operator. Finding a decomposition of this unitary into elementary quantum gates (such as single-qubit gates and CNOTs) is generally non-trivial and often depends on the structure and properties of the target matrix. This challenge motivates the development of algorithmic strategies tailored to different matrix classes and decomposition schemes.

We explore various strategies for constructing explicit block encodings. In particular, we review techniques for performing algebraic operations—such as matrix addition and multiplication—on block-encoded operators. These methods are especially useful when combining different techniques arising from matrix decompositions or structural insights. We also present an algorithm for block encoding a generic matrix by explicitly encoding each of its elements, and we introduce a compression method that reduces the circuit size while maintaining a desired accuracy threshold [10]. Furthermore, we discuss a block encoding approach tailored to sparse matrices—i.e., matrices with only a few non-zero entries [11]. Finally, we provide an explicit circuit construction for block encoding circulant matrices, a structured class of matrices with efficient quantum representations.

Building upon these foundational strategies for constructing block encodings, we now turn to the *Linear Combination of Unitaries* (LCU) technique, a powerful method for representing non-unitary matrices as linear combinations of efficiently implementable unitary operators. The LCU framework enables block encoding of such operators by preparing a quantum superposition over control states and coherently applying a controlled selection of unitaries. The general construction involves three quantum oracles: a black-box quantum operation that performs a well-defined transformation. P_R prepares the coefficients of the linear combination in superposition; SELECT applies the appropriate unitary conditioned on the control register; and P_L uncomputes the initial state preparation to ensure coherence.

The main contribution of this thesis is a novel and more efficient variant of this framework, which we call Fast One-Qubit Control Select LCU (FOQCS-LCU). This new formulation significantly reduces the computational cost of block encoding circuits by an order of magnitude in practice and also asymptotically; while maintaining the same high-level structure of the LCU method. Additionally, by exploiting the structural properties of the target Hamiltonians, we introduce a new class of efficient Dicke state- a quantum state consisting of a superposition of all computational basis states with a fixed number of qubits in the $|1\rangle$ state [12]- preparation routines that substantially reduce the cost of the P_R and P_L oracles. These improvements enable an order-of-magnitude reduction in the total number of gates required, providing a practical advantage for quantum simulation and related applications.

To validate our approach, we construct explicit block encoding circuits for the Heisenberg model. The Heisenberg model is a central object of study in quantum many-body physics. It describes interacting spins- $\frac{1}{2}$ arranged on a lattice and is defined by a Hamiltonian that is naturally expressed as a sum of tensor products of Pauli operators. This formulation aligns directly with the quantum computing framework, where qubits represent spin- $\frac{1}{2}$ particles: the computational basis states $|0\rangle$ and $|1\rangle$ correspond to the spin-up and spin-down states, respectively. Because of this direct correspondence,

the Heisenberg Hamiltonian can be efficiently encoded in a quantum circuit using standard Pauli operations. We compute detailed, non-asymptotic gate counts and conduct numerical benchmarks comparing our FOQCS-LCU method with standard LCU and other known techniques. The results demonstrate that FOQCS-LCU achieves an order-of-magnitude reduction in CNOT gate count, highlighting its practical benefits for Hamiltonian simulation and related quantum applications.

Chapter 2

Preliminaries on quantum computing

For the theoretical and technical foundations used throughout this chapter, we refer to several key sources. Specifically, [13] and [14] are lecture notes that provide a clear and modern treatment of quantum computing. The references [1] and [15] are textbooks on quantum computation offering both theoretical background and practical insights. Finally, the article [16] presents an introduction of quantum computing from a mathematical perspective.

2.1 Quantum Bits

In classical computing, the bit, short for binary digit, is the basic unit of information. A bit can exist in one of two distinct states, typically labeled as 0 and 1, and can be physically realized using any system with two stable states. There exist an analogous concept in quantum computation: the qubit – short for quantum bit- is a mathematical representation of a two – state quantum-mechanical system.

Definition 2.1.1. The vectors

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

are called the *basis states* of a quantum bit. Both vectors belong to the Hilbert space \mathbb{C}^2 .

The difference between bits and qubits is that a qubit can also be in a state other than $|0\rangle$ and $|1\rangle$. Its generic state is a linear combination over the complex numbers of both basis states.

Definition 2.1.2. A pure qubit state $|\psi\rangle$ is a unit vector that is a linear combination of the basis states:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

where the coefficients $\alpha, \beta \in \mathbb{C}$ are called the amplitudes of the state. Moreover proportional vectors represent the same quantum state, as global phases have no physical significance in quantum mechanics.

The notation $|\rangle$ is known as *Dirac notation*, and it is the standard formalism for representing quantum states. Specifically, $|\psi\rangle$ denotes a column vector (a state vector), while $\langle\psi|$ represents its conjugate transpose (also called the dual vector). This allows us to write expressions such as:

$$\langle\psi|\psi\rangle = \|\psi\|^2,$$

where the inner product $\langle\psi|\psi\rangle$ gives the squared norm of the vector ψ in the Euclidean (or 2-) norm.

The vectors $|0\rangle$ and $|1\rangle$ form an orthonormal basis of \mathbb{C}^2 . From this point onward, this basis will be referred to as the *computational basis* of a qubit. However, other bases are also commonly used to represent qubit states. One

such example is the *Hadamard basis*, defined by:

$$\begin{aligned} |+\rangle &:= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ |-\rangle &:= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{aligned}$$

We remark that the fact that $|\psi\rangle$ is a unit vector implies that $|\alpha|^2 + |\beta|^2 = 1$.

1. So we can rewrite $|\psi\rangle$ as :

$$|\psi\rangle = e^{i\gamma} \left(\cos \frac{\theta}{2} |1\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle \right), \quad \theta, \phi, \gamma \in \mathbb{R}.$$

Ignoring the irrelevant global phase γ , the state is effectively:

$$|\psi\rangle = \cos \frac{\theta}{2} |1\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle, \quad 0 \leq \theta \leq \pi, \quad 0 \leq \phi < 2\pi.$$

Thus, each single-qubit quantum state can be uniquely represented by a point on the unit three-dimensional sphere, known as the Bloch sphere (fig. 2.1), as:

$$\mathbf{a} = (\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta).$$

The Bloch sphere is a useful tool for visualizing the state of a single qubit and serves as an effective testbed for concepts in quantum computation and quantum information. Many single-qubit operations can be naturally described within this framework [1].

Extending from a single qubit to a multi-qubit system naturally involves constructing a Hilbert space that combines multiple copies of \mathbb{C}^2 . This is achieved by taking the tensor product of the individual qubit spaces. Let us first introduce the concept by explaining the case of two qubits. If these were two classical bits, there would be four possible states: 00, 01, 10, and 11. Analogously, a two-qubit quantum system has four computational basis states, denoted $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. Unlike classical bits, however, a pair of qubits can exist in a superposition of these basis states. The general quantum state of two qubits is described by a state vector of the form

$$|\psi\rangle = \alpha_{00} |00\rangle + \alpha_{01} |01\rangle + \alpha_{10} |10\rangle + \alpha_{11} |11\rangle,$$

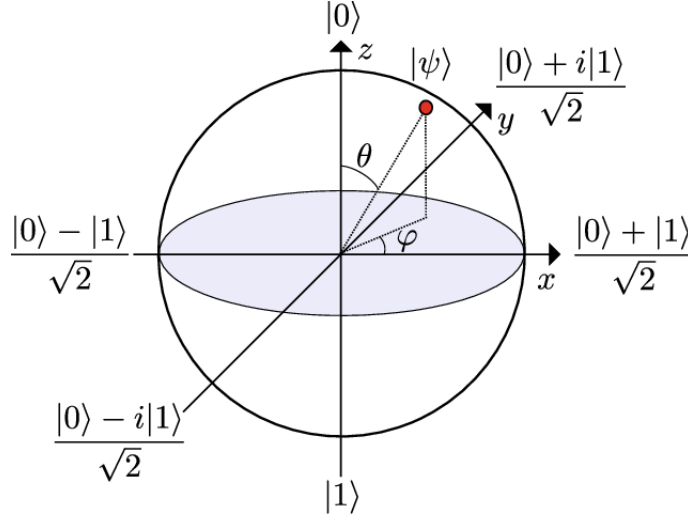


Figure 2.1: Representation of a single-qubit pure state $|\psi\rangle$ on the Bloch sphere.

where each α_{ij} is a complex amplitude. As in the single-qubit case, a measurement yields one of the basis states $|x\rangle$ (with $x \in \{00, 01, 10, 11\}$) with probability $|\alpha_x|^2$, and the state collapses to $|x\rangle$ after the measurement. The normalization condition,

$$\sum_{x \in \{0,1\}^2} |\alpha_x|^2 = 1,$$

ensures that the total probability sums to one.

After this explanation of how to describe a two-qubit system, it is necessary to introduce the tensor product of two Hilbert spaces to continue the introduction to multi-qubit systems. For this part, we refer to [16].

Definition 2.1.3. Let V and W be two vector spaces over the field F . One consider first a vector space L that has the Cartesian product $V \times W$ as a basis. That is, the basis elements of L are the pairs (v, w) with $v \in V$ and $w \in W$. To get the tensor product of V and W , one can define it as the vector space of the functions $V \times W \rightarrow F$ that have a finite number of nonzero values and identifying (v, w) with the function that takes the value 1 on (v, w) and 0 otherwise. Let R be the linear subspace of L that is spanned

by the elements of one of the forms:

$$\begin{aligned} (v_1 + v_2, w) - (v_1, w) - (v_2, w), \\ (v, w_1 + w_2) - (v, w_1) - (v, w_2), \\ (\alpha v, w) - \alpha(v, w), \\ (v, \alpha w) - \alpha(v, w), \end{aligned}$$

where $v, v_1, v_2 \in V$, $w, w_1, w_2 \in W$ and $\alpha \in F$. Then, the tensor product is defined as the quotient space:

$$V \otimes W = L/R,$$

and the image of (v, w) in this quotient is denoted $v \otimes w$.

A related definition is the concept of the tensor product between linear operators.

Definition 2.1.4. Let A and B be linear operators defined on V and W respectively, then the linear operator $A \otimes B$ operating on $V \otimes W$ is the unique linear operator on $V \otimes W$ such that :

$$(A \otimes B)(v \otimes w) = Av \otimes Bw$$

with $v \in V$ and $w \in W$.

If A and B are $n \times m$ and $\ell \times p$ matrices, respectively, that represent linear operators A and B with respect to the canonical basis, then the linear operator $A \otimes B$ (called the **tensor product** or **Kronecker product** of A and B) has the following matrix representation with respect to the canonical basis:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1m}B \\ a_{21}B & a_{22}B & \cdots & a_{2m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}B & a_{n2}B & \cdots & a_{nm}B \end{bmatrix},$$

That is, each entry a_{ij} of matrix A is replaced by the scalar multiple $a_{ij}B$, resulting in an $n\ell \times mp$ matrix.

The definition of the inner product on a tensor product space arises naturally from the inner product on the constituent spaces:

Definition 2.1.5. Let V and W be Hilbert spaces with inner products $\langle \cdot, \cdot \rangle_V$ and $\langle \cdot, \cdot \rangle_W$, respectively. On this tensor product, we can define an inner product by setting

$$\langle v_1 \otimes w_1, v_2 \otimes w_2 \rangle := \langle v_1, v_2 \rangle_V \langle w_1, w_2 \rangle_W$$

for $v_1, v_2 \in V$ and $w_1, w_2 \in W$.

In particular, note that the tensor product of unit vectors is itself a unit vector. The tensor product is fundamental for describing composite quantum systems, as it allows us to represent the joint state of multiple qubits within a single vector space.

Example 2.1.1. If $|0\rangle$ and $|1\rangle$ are the basis states of a quantum bit, the tensor product $|0\rangle \otimes |1\rangle$ is given by:

$$|0\rangle \otimes |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Remark 2.1.1. Given two bases for vector spaces V and W , say $\mathcal{B}_V = \{v_1, \dots, v_n\}$, $\mathcal{B}_W = \{w_1, \dots, w_m\}$, respectively, the set

$$\mathcal{B}_{V \otimes W} = \{v_i \otimes w_j \mid 1 \leq i \leq n, 1 \leq j \leq m\}$$

is a basis of the tensor product space $V \otimes W$.

In our setting, these corresponding bases will be particularly useful for representing integers. On a classical computer, a non-negative integer $a \in \mathbb{N}$ such that $a < 2^n$ (i.e., it can be described using n bits) is typically expressed in base-2 as:

$$a = \sum_{\ell=0}^{n-1} a_\ell 2^\ell, \quad \text{where } a_\ell \in \{0, 1\} \text{ are the binary digits of } a. \quad (2.1)$$

On a quantum computer, we can represent an integer $a < 2^n$ using n qubits as:

$$|a\rangle_n = |a_0 a_1 \cdots a_{n-1}\rangle = \bigotimes_{\ell=0}^{n-1} |a_\ell\rangle .$$

For example, the number 27 can be represented with 5 qubits (since $29 < 2^5$) as:

$$|27\rangle_5 = |11011\rangle = |1\rangle \otimes |1\rangle \otimes |0\rangle \otimes |1\rangle \otimes |1\rangle .$$

In this way, integers are always represented by elements of the basis obtained from tensor products of the single-qubit computational basis vectors. This basis is subsequently referred to as the *computational basis*. In particular, we remark that:

$$|2^i\rangle_n = |\underbrace{0 \cdots 0}_i 1 \underbrace{0 \cdots 0}_{n-i-1}\rangle$$

Now we are ready to define a quantum state of a generic n -qubit system.

Definition 2.1.6 ([16] Def.3.5). The state $|\psi\rangle_n$ of a generic n -qubit system is a superposition (that is, a linear combination) of the 2^n states of the computational basis $|0\rangle_n, \dots, |2^n - 1\rangle_n$ with modulus 1. In particular,

$$|\psi\rangle_n = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle_n ,$$

with amplitudes $\alpha_j \in \mathbb{C}$ constrained to

$$\sum_{j=0}^{2^n-1} |\alpha_j|^2 = 1 .$$

In the following, we will simply write $|\psi\rangle$ instead of $|\psi\rangle_n$ when the value of n is clear from the context. In a multi-qubit quantum system, a quantum state can be either a *product state* or an *entangled state*. A product state is one that can be written as a tensor product of individual states. For example, a two-qubit state $|\psi\rangle$ is a product state if there exist single-qubit states $|\psi_1\rangle$ and $|\psi_2\rangle$ such that:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle .$$

For example:

$$|\psi\rangle = |0\rangle \otimes |1\rangle = |01\rangle .$$

In this case, the qubits are uncorrelated. An entangled state is a state that *cannot* be written as a product of individual qubit states. That is, there do not exist single-qubit states $|\psi_1\rangle$ and $|\psi_2\rangle$ such that:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$$

A classic example of an entangled state is the Bell state:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \tag{2.2}$$

This state cannot be separated into a tensor product of two single-qubit states. Measurements on one qubit instantaneously affect the outcome probabilities on the other as we will see later.

2.2 Quantum measurement

Quantum mechanics imposes fundamental limits on the amount of information that can be extracted from a quantum state. When measuring a qubit in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the result is either 0, with probability $|\alpha|^2$, or 1, with probability $|\beta|^2$. Naturally, these probabilities must sum to one: $|\alpha|^2 + |\beta|^2 = 1$, which is why $|\psi\rangle$ is a unit vector. Even though a single qubit can exist in infinitely many possible states (you can imagine this using the Bloch sphere), this doesn't mean it can store an infinite amount of information. When we measure a qubit, we only get one classical bit of output—either a 0 or a 1. And once we measure it, the qubit's original state is lost, as the measurement causes it to “collapse” into one of those two outcomes. A more subtle question is: how much information does a qubit hold when we don't measure it? This might seem strange, since unmeasured information can't be directly observed. However, during quantum evolution (when the qubit changes according to unitary operations), it seems that Nature keeps track of the exact details of the qubit's state. In this way, we can

say the qubit holds a lot of "hidden information." But what does it actually mean to measure a quantum state? To understand this, we need to be more precise. In quantum mechanics, a measurement isn't just checking the value of a variable like in classical physics. Instead, we measure a quantum system with respect to a special kind of object called a quantum observable. This is where the formal framework of quantum mechanics begins to play a key role. Quantum observables correspond to physical quantities that can be measured from a quantum system, such as energy, position, or spin. Here, we focus on their mathematical description.

Definition 2.2.1. A quantum observable always corresponds to a Hermitian matrix M , which has the spectral decomposition:

$$M = \sum_m \lambda_m P_m ,$$

where $\lambda_m \in \mathbb{R}$ are the eigenvalues of M , and P_m are the projection operators into the eigenspaces associated with λ_m , i.e, $P_m^2 = P_m$.

Hermitian operators have real eigenvalues and a complete set of orthonormal eigenvectors, making them suitable for modeling measurable physical quantities. When a quantum state $|\psi\rangle$ is measured by a quantum observable M , the outcome of the measurement is always an eigenvalue λ_m with probability:

$$p_m = \langle \psi | P_m | \psi \rangle .$$

After the measurement, the quantum state becomes :

$$|\psi\rangle \rightarrow \frac{P_m |\psi\rangle}{\sqrt{p_m}} .$$

The projector operators satisfy the completeness equation:

$$\sum_m P_m = I .$$

The completeness equation expresses the fact that probabilities sum to one:

$$\sum_m p_m = \sum_m \langle \psi | P_m | \psi \rangle = \langle \psi | \left(\sum_m P_m \right) | \psi \rangle = \langle \psi | \psi \rangle = 1 .$$

In quantum computing, we will measure our state by the following quantum observable:

$$M = \sum_{m=0}^{2^n-1} m \cdot |m\rangle \langle m|$$

In this case, the probability of measuring m will be:

$$p_m = \langle \psi | m \rangle \langle m | \psi \rangle = \alpha_m^* \alpha_m = |\alpha_m|^2$$

It is also possible to measure just one part of the state. Let

$$M = 0 \cdot |0\rangle \langle 0| \otimes I + 1 \cdot |1\rangle \langle 1| \otimes I$$

be a quantum observable. Suppose we measure a quantum state of the form

$$|\psi\rangle = |0\rangle |\psi_0\rangle + |1\rangle |\psi_1\rangle.$$

Then, the probability of obtaining the outcome 0 is $\| |\psi_0\rangle \|^2$, and the probability of obtaining the outcome 1 is $\| |\psi_1\rangle \|^2$. If the measurement yields 0, then the post-measurement (collapsed) state is

$$|0\rangle \otimes \frac{|\psi_0\rangle}{\| |\psi_0\rangle \|},$$

and similarly, if the outcome is 1, the resulting state is

$$|1\rangle \otimes \frac{|\psi_1\rangle}{\| |\psi_1\rangle \|}.$$

This type of measure is very important when we have entangled qubit state, like, for example, the Bell state in eq. (2.2).

Example 2.2.1. Suppose we perform a measurement only on the first qubit, in the computational basis $\{|0\rangle, |1\rangle\}$. To understand the effect of this measurement, we rewrite the state:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} (|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$$

This expression shows that:

- With probability $\frac{1}{2}$, we obtain the outcome $|0\rangle$ on the first qubit. In this case, the full system collapses to:

$$|0\rangle \otimes |0\rangle = |00\rangle ;$$

- With probability $\frac{1}{2}$, we obtain the outcome $|1\rangle$ on the first qubit. Then, the full system collapses to:

$$|1\rangle \otimes |1\rangle = |11\rangle .$$

In both cases, the second qubit becomes *perfectly correlated* with the measurement outcome of the first qubit. That is, if we measure $|0\rangle$ on the first qubit, the second qubit is guaranteed to be in $|0\rangle$; if we measure $|1\rangle$, the second qubit is $|1\rangle$.

2.3 Quantum Circuit

The quantum circuit model of computation represents quantum information using wires and gates. Quantum information propagates along the wires, while gates—placed at specific points on the wires—act to transform the quantum state before passing it along the circuit. The *quantum circuit language* provides a graphical and compact way to represent the application of a sequence of quantum operators to a quantum state. The operator applied to a qubit is represented by a matrix that must preserve the norm of quantum states, ensuring that the state remains normalized (i.e., with norm equal to 1). Therefore, such an operator must be unitary.

Definition 2.3.1 (Def 4.4 [16]). A quantum gate operating on a space of one qubit is represented by a unitary matrix $U \in U(2)$. More generally, a quantum gate acting on an n -qubit system is represented by a matrix $U \in U(2^n)$.

In the quantum circuit language, time flows from left to right: the input quantum state appears on the left, and each “wire” represents a qubit:

$$|\psi\rangle \text{ --- } \boxed{U} \text{ --- } U|\psi\rangle$$

Some of the most important single-qubit gates are the Pauli matrices:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

the Pauli matrices give rise to three useful classes of unitary matrices: the rotation operators are defined as follows:

$$\begin{aligned} R_x(\theta) &= \begin{bmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix}, \\ R_y(\theta) &= \begin{bmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix}, \\ R_z(\theta) &= \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}. \end{aligned}$$

Other important quantum gates are the Hadamard gate (denoted by H), the S -gate and the T -gate:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$$

Let us now present some examples of simple quantum circuits:

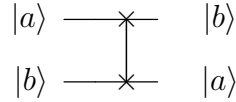
$$\begin{aligned} |a\rangle &\text{ --- } \boxed{H} \text{ --- } \frac{1}{\sqrt{2}} |0\rangle + \frac{(-1)^a}{\sqrt{2}} |1\rangle \\ |a\rangle &\text{ --- } \boxed{X} \text{ --- } |1 \oplus a\rangle \\ |0\rangle &\text{ --- } \boxed{R_y(\theta)} \text{ --- } \cos \frac{\theta}{2} |0\rangle + \sin \frac{\theta}{2} |1\rangle \end{aligned}$$

In order to construct more sophisticated quantum circuits, we introduce the two-qubit gate known as the CNOT (controlled-NOT) gate, which is defined as follows:

$$\begin{array}{c} |a\rangle \text{ --- } \bullet \text{ --- } |a\rangle \\ |b\rangle \text{ --- } \oplus \text{ --- } |a \oplus b\rangle \end{array}$$

Where the "dot" means that the quantum gate connected to the dot only becomes active if the state of the first qubit, called the control qubit, is $a = 1$. This is the reason of the name of the CNOT gate (controlled NOT).

Another two-qubit gate used is the SWAP gate, which swaps the state of two qubits:



By combining CNOT gates with single-qubit gates, it is possible to construct any unitary transformation on an arbitrary number of qubits. This property makes the set of CNOT and single-qubit gates *universal*. In quantum computing, a set of gates is said to be universal if any unitary operation –and thus any quantum algorithm –can be decomposed into a finite sequence of gates from that set. The decomposition of a quantum operation into elementary gates is not unique, and finding a decomposition that minimizes the number of qubits is a critical aspect of analyzing the computational cost of quantum algorithms. Moreover, we introduce the following definition.

Definition 2.3.2. The *depth* of a quantum circuit is defined as the maximum number of sequential steps required to execute all operations, from the initial state preparation to the final measurement. Formally, it corresponds to the length of the longest directed path through the circuit, where the path follows the temporal order of gates along the qubit wires.

Intuitively, the circuit depth represents the minimum number of discrete time steps needed to implement the circuit, under the assumptions that:

- gates acting on disjoint sets of qubits can be applied simultaneously,
- gates sharing at least one qubit must be executed at different time steps and in the correct order.

In general, when evaluating the complexity of a quantum circuit, the following factors are typically considered:

- **The number of CNOT gates.** These two-qubit gates are generally more error-prone and resource-intensive to implement on current quantum hardware compared to single-qubit gates.

- **The depth of the circuit.** It serves as a measure of the circuit's runtime and is directly related to susceptibility to decoherence and time-dependent noise.
- **The number of qubits used.** The qubit count is a key resource metric, especially on near-term devices where qubit availability is limited and increasing the number of qubits may lead to higher error rates due to noise, cross-talk, and hardware constraints.

In many quantum algorithms, we make use of oracles, which are abstract black-box operations that encode information about a problem instance. Formally, an oracle is a unitary operator that performs a specific transformation on quantum states, often depending on some hidden function or data. To denote such operators, we may use the notation U for an n -qubit gate acting on the state space \mathbb{C}^{2^n} .

$$\text{---} \overset{n}{\boxed{U}} \text{---}$$

In particular, it is common to use the following notation to represent a controlled gate:

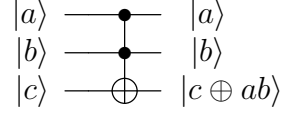
$$\text{---} \bullet \text{---} \text{---} \boxed{U} \text{---}$$

This denotes that the unitary gate U is applied to the target qubit if and only if the control qubit is in the state $|1\rangle$. It also may happen that we want to apply the gate only if the controlled gate is in the state $|0\rangle$. In this case we write:

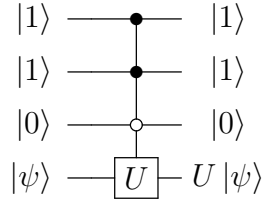
$$\text{---} \circ \text{---} \text{---} \boxed{U} \text{---}$$

Such notation is especially useful for improving the readability of quantum circuits, as it allows one to abstract away the full decomposition of the controlled operation while maintaining a clear and concise circuit representation. It is also possible for a quantum gate to be controlled by more than one qubit, leading to multi-controlled operations. For example, one can use a controlled-CNOT gate with two control qubits, commonly known as the Toffoli gate. This gate applies a NOT operation to the target qubit if and only if both

control qubits are in the state $|1\rangle$.

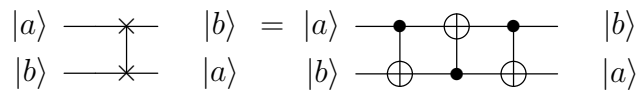


More generally, a multi-controlled gate is an operation that applies a target unitary U conditioned on the state of multiple control qubits. Specifically, the gate acts as the identity unless all control qubits are in the state $|1\rangle$, in which case the unitary U is applied to the target qubit or register. These gates are often denoted as $C^n(U)$, where n is the number of control qubits:



Multi-controlled gates are very used in quantum algorithms since they can be conceptually simple; however their implementation on real hardware is not trivial since they need a decomposition in one and two qubit gates. In the following we present some examples of decomposition of qubits gates into sequences of CNOT and single-qubit gates.

Example 2.3.1. The SWAP gate can be implemented by 3 CNOT gates:



After the first CNOT gate, the state becomes:

$$|a\rangle |a \oplus b\rangle ,$$

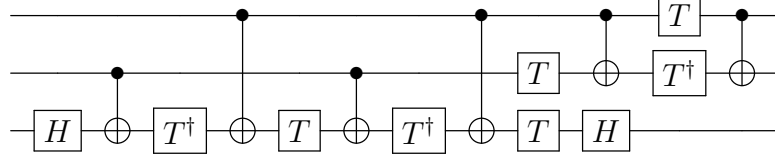
then, after the second CNOT gate, we get:

$$|a \oplus a \oplus b\rangle |a \oplus b\rangle ,$$

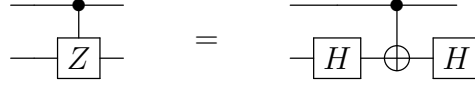
where $a \oplus a \oplus b$ is equal to b . Finally, after the third one, we have:

$$|b\rangle |a \oplus b \oplus b\rangle = |b\rangle |a\rangle .$$

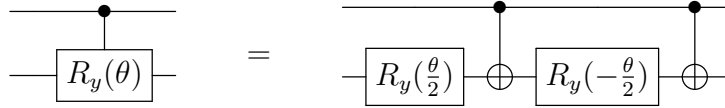
Example 2.3.2 ([1] Figure 4.9). The Toffoli gate can be implemented by 6 CNOT with the following circuit:



Example 2.3.3. The controlled Z gate can be implemented by 1 CNOT and 2 Hadamard gate:



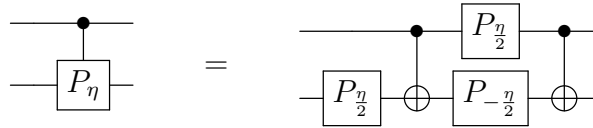
Example 2.3.4. The controlled rotation y gate ($C\text{-}R_y(\theta)$) can be implemented by 2 CNOTs with the following circuit:



Example 2.3.5. The phase gate P_η is defined as:

$$P_\eta = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\eta} \end{bmatrix},$$

for a generic $\eta \in [0, 2\pi]$. The controlled phase gate can be implemented by 2 CNOTs with the following circuit:



Chapter 3

Quantum singular value transformation

Drawing on the works of Martyn et al. [4] and Gilyén et al. [3], this chapter provides an overview of the framework of the quantum eigenvalue transformation algorithm. It enables the application of polynomial transformations to the eigenvalues of a linear operator embedded within a unitary matrix.

The Quantum Singular Value Transformation (QSVT) generalizes the quantum eigenvalue transform to the case of non-Hermitian operators, enabling polynomial transformations of singular values rather than just eigenvalues. This broader framework forms the foundation of many recent advances in quantum linear algebra and algorithm design. In this chapter, however, we develop the theory only up to Quantum Eigenvalue Transformation, as this is sufficient for our purposes. Since we will be working exclusively with hermitian matrices, it is not necessary to consider the more general QSVT framework in detail.

These algorithms are particularly significant because they provide a unifying framework that connects several foundational quantum algorithms, including quantum search, quantum phase estimation, and Hamiltonian simulation. These three algorithms are among the most powerful demonstrations of quantum speed-up over classical computation and form the core arguments

in favor of the potential advantages of quantum computers.

3.1 Quantum signal processing

Quantum Signal Processing (QSP) amounts to transforming an input signal through a sequence of quantum operations. The signal is represented by a matrix known as the *signal-rotation operator*. A typical example of such an operator is:

$$W(a) = \begin{bmatrix} a & i\sqrt{1-a^2} \\ -i\sqrt{1-a^2} & a \end{bmatrix}, \quad (3.1)$$

which is a Pauli x -rotation by the angle $\theta = 2 \arccos a$. This unitary matrix rotates the quantum state in a way that depends on the signal parameter a . The aim of QSP is to modify the signal a by a polynomial transformation. In order to do that, we alternate the operator W with an operator $S(\phi)$ known as *signal processing operator*:

$$S(\phi) = e^{i\phi Z}, \quad (3.2)$$

with $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. We remark that $S(\phi)$ is a Pauli z -rotation by an angle 2ϕ . Consider a sequence of phases $\vec{\phi} = (\phi_0, \phi_1, \dots, \phi_d) \in \mathbb{R}^{d+1}$. The QSP operation sequence $U_{\vec{\phi}}$ is defined as

$$U_{\vec{\phi}}(a) = e^{i\phi_0 Z} \prod_{k=1}^d \left(W(a) e^{i\phi_k Z} \right). \quad (3.3)$$

The QSP sequences can produce signal transformations, since the matrix element

$$P(a) = \langle 0 | U_{\vec{\phi}}(a) | 0 \rangle. \quad (3.4)$$

is a polynomial function of a , whose degree is at most equal to the length of the QSP phase sequence $\vec{\phi}$. For example, for $\vec{\phi} = (0, 0)$, we have $P(a) = a$; for $\vec{\phi} = (0, 0, 0)$, $P(a) = 2a^2 - 1$; and for $\vec{\phi} = (0, 0, 0, 0)$, $P(a) = 4a^3 - 3a$. These correspond to the Chebyshev polynomials of the first kind, and the pattern continues accordingly. Specifically, we have the following theorem:

Theorem 3.1.1 ([17]). The QSP sequence $U_{\vec{\phi}}$ produces a matrix that can be expressed as a polynomial function of a :

$$e^{i\phi_0 Z} \prod_{k=1}^d W(a) e^{i\phi_k Z} = \begin{bmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ iQ^*(a)\sqrt{1-a^2} & P^*(a) \end{bmatrix}, \quad (3.5)$$

for $a \in [-1, 1]$, and a $\vec{\phi}$ exists for any polynomials P, Q in a such that:

1. $\deg(P) \leq d$, $\deg(Q) \leq d - 1$,
2. P has parity $d \bmod 2$ and Q has parity $(d - 1) \bmod 2$,
3. $|P|^2 + (1 - a^2)|Q|^2 = 1$.

Proof. “ \implies ”: For the $k = 0$ case the unitary on the left hand side of eq. (3.5) is $e^{i\phi_0 Z}$, so that $P \equiv e^{i\phi_0}$ and $Q \equiv 0$ satisfy the properties. Now we prove the statement by induction. Suppose we have proved for $d - 1$ that

$$e^{i\phi_0 Z} \prod_{k=1}^{d-1} W(a) e^{i\phi_k Z} = \begin{bmatrix} \tilde{P}(a) & i\tilde{Q}(a)\sqrt{1-a^2} \\ i\tilde{Q}^*(a)\sqrt{1-a^2} & \tilde{P}^*(a) \end{bmatrix},$$

where \tilde{P}, \tilde{Q} are polynomials which satisfy 1-3. Then

$$\begin{aligned} U_{\vec{\phi}}(a) &= e^{i\phi_0 Z} \prod_{k=1}^{d-1} W(a) e^{i\phi_k Z} \\ &= \begin{bmatrix} \tilde{P}(a) & i\tilde{Q}(a)\sqrt{1-a^2} \\ i\tilde{Q}^*(a)\sqrt{1-a^2} & \tilde{P}^*(a) \end{bmatrix} \begin{bmatrix} e^{i\phi_k} a & ie^{-i\phi_k} \sqrt{1-a^2} \\ ie^{i\phi_k} \sqrt{1-a^2} & e^{-i\phi_k} a \end{bmatrix} \\ &= \begin{bmatrix} e^{i\phi_k} (a\tilde{P}(a) + (a^2 - 1)\tilde{Q}(a)) & ie^{-i\phi_k} (a\tilde{Q}(a) + \tilde{P}(a))\sqrt{1-a^2} \\ ie^{i\phi_k} (a\tilde{Q}^*(a) + \tilde{P}^*(a))\sqrt{1-a^2} & e^{-i\phi_k} (a\tilde{P}^*(a) + (a^2 - 1)\tilde{Q}^*(a)) \end{bmatrix}, \end{aligned} \quad (3.6)$$

we define $P(a) := e^{i\phi_k} (a\tilde{P}(a) + (a^2 - 1)\tilde{Q}(a))$ and $Q(a) := e^{-i\phi_k} (a\tilde{Q}(a) + \tilde{P}(a))$. These polynomials satisfy condition 1, since by the inductive hypothesis we have $\deg(\tilde{P}) \leq d - 1$ and $\deg(\tilde{Q}) \leq d - 2$. Thus,

$$\deg(a\tilde{P}(a)) \leq d, \quad \deg((a^2 - 1)\tilde{Q}(a)) \leq d,$$

and so $\deg(P) \leq d$. Similarly, $\deg(Q) \leq d - 1$. Condition 2 is also satisfied. Since $\tilde{P}(a)$ has parity $d - 1 \bmod 2$, the term $a\tilde{P}(a)$ has parity $d \bmod 2$; likewise, both $a^2\tilde{Q}(a)$ and $\tilde{Q}(a)$ have parity $d \bmod 2$, so $P(a)$ has parity $d \bmod 2$. For $Q(a)$, we note that $a\tilde{Q}(a)$ has parity $d - 1 \bmod 2$ and $\tilde{P}(a)$ has parity $d - 1 \bmod 2$, hence $Q(a)$ has parity $d - 1 \bmod 2$, as required. Finally note that the left hand side of eq. (3.5) is a product of unitaries, therefore the right hand side is unitary too, which implies 3.

" \Leftarrow ": Suppose $P(a), Q(a)$ satisfy properties 1–3. We first handle the trivial case: if $\deg(P) = 0$, then condition 3 implies $|P(1)| = 1$, so $P(a) \equiv e^{i\phi_0}$ for some $\phi_0 \in \mathbb{R}$, and hence $Q(a) \equiv 0$. Since Q must have parity $(d - 1) \bmod 2$, it follows that d must be even. In this case, we may take

$$\vec{\phi} = (\phi_0, \frac{\pi}{2}, \dots, \frac{\pi}{2}) \in \mathbb{R}^{d+1},$$

which yields the desired sequence. This case also serves as the base for our induction.

Now, assume the result holds for degree $d - 1$, and let us prove the step for d . Condition 3 can be rewritten as:

$$\forall a \in [-1, 1] : \quad P(a)P^*(a) + (1 - a^2)Q(a)Q^*(a) = 1.$$

Since this equation holds on an infinite set, it must be an identity of polynomials. Suppose $\deg(P) = \ell \in [1, d]$, then necessarily $\deg(Q) = \ell - 1$ and $|p_\ell| = |q_{\ell-1}|$ so that the highest-degree terms cancel out in the above expression.

Let $\phi_d \in \mathbb{R}$ be such that $e^{2i\phi_d} = \frac{p_\ell}{q_{\ell-1}}$. Define:

$$\begin{aligned} & \begin{bmatrix} \tilde{P}(a) & i\tilde{Q}(a)\sqrt{1-a^2} \\ i\tilde{Q}^*(a)\sqrt{1-a^2} & \tilde{P}^*(a) \end{bmatrix} := \begin{bmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ iQ^*(a)\sqrt{1-a^2} & P^*(a) \end{bmatrix} e^{-i\phi_d Z} W^\dagger(a) \\ &= \begin{bmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ iQ^*(a)\sqrt{1-a^2} & P^*(a) \end{bmatrix} \begin{bmatrix} e^{-i\phi_d} a & -ie^{-i\phi_d}\sqrt{1-a^2} \\ -ie^{i\phi_d}\sqrt{1-a^2} & e^{i\phi_d} a \end{bmatrix} \\ &= \begin{bmatrix} \tilde{P}(a) & i\tilde{Q}(a)\sqrt{1-a^2} \\ i\tilde{Q}^*(a)\sqrt{1-a^2} & \tilde{P}^*(a) \end{bmatrix}, \end{aligned}$$

where:

$$\begin{aligned}\tilde{P}(a) &= e^{-i\phi_d} a P(a) + e^{i\phi_d} (1 - a^2) Q(a) = e^{-i\phi_d} \left(a P(a) + \frac{p_\ell}{q_{\ell-1}} (1 - a^2) Q(a) \right), \\ \tilde{Q}(a) &= e^{i\phi_d} a Q(a) - e^{-i\phi_d} P(a) = e^{-i\phi_d} \left(\frac{p_\ell}{q_{\ell-1}} a Q(a) - P(a) \right).\end{aligned}$$

The highest-order terms cancel, so $\deg(\tilde{P}) \leq \ell - 1 \leq d - 1$ and $\deg(\tilde{Q}) \leq \ell - 2 \leq d - 2$. Thus, \tilde{P}, \tilde{Q} satisfy conditions 1 and 2 for degree $d - 1$. Condition 3 is preserved because $W^\dagger(a)$ and $e^{-i\phi_d Z}$ are unitary transformations.

By the induction hypothesis, there exists $\vec{\phi} \in \mathbb{R}^d$ such that:

$$\begin{bmatrix} \tilde{P}(a) & i\tilde{Q}(a)\sqrt{1-a^2} \\ i\tilde{Q}^*(a)\sqrt{1-a^2} & \tilde{P}^*(a) \end{bmatrix} = U_{\vec{\phi}}(a).$$

It follows that the original $U_{\vec{\phi}}(a)$ is given by appending ϕ_d :

$$U_{\vec{\phi}}(a) = U_{\vec{\phi}}(a) W(a) e^{i\phi_d Z},$$

where $\vec{\phi} := (\tilde{\phi}_0, \tilde{\phi}_1, \dots, \tilde{\phi}_{d-1}, \phi_d) \in \mathbb{R}^{d+1}$.

Hence, the claim holds for all d . \square

When we define $P(a) = \langle 0 | U_{\vec{\phi}} | 0 \rangle$, we are constrained to polynomials P for which there exists a corresponding polynomial Q satisfying the conditions of theorem 3.1.1. This requirement can be restrictive in practice. For instance, when $a = \pm 1$, the entire QSP sequence reduces to a single z -rotation, which implies that $|P(\pm 1)| = 1$. Consequently, the class of admissible polynomials is limited to those that satisfy this boundary constraint. To address this limitation, we consider an alternative observable:

$$\text{Poly}(a) = \langle + | U_{\vec{\phi}} | + \rangle = \text{Re}[P(a)] + i \text{Re}[Q(a)] \cdot \sqrt{1 - a^2}.$$

This transformation expands the class of achievable polynomial transformations. In particular, it can be shown [4] that using this construction, one can approximate any real-valued polynomial $\text{Poly}(a)$ with parity $d \bmod 2$, degree at most d , and bounded in magnitude by 1 over the interval $[-1, 1]$, i.e., $|\text{Poly}(a)| \leq 1$ for all $a \in [-1, 1]$. This is achieved by selecting an appropriate P such that $\text{Re}[P(a)]$ approximates the desired function, and a Q with a sufficiently small real component to control the imaginary part.

3.2 Amplitude Amplification

An application of Theorem 3.1.1 is the following problem, known as amplitude amplification. Suppose we are given a unitary operator U , its inverse U^\dagger , and two phase oracle operators A_ϕ and B_ϕ , each of which applies a phase shift to a specific, privileged state:

$$A_\phi = e^{i\phi|A_0\rangle\langle A_0|}, \quad (3.7)$$

$$B_\phi = e^{i\phi|B_0\rangle\langle B_0|}. \quad (3.8)$$

The objective is to construct a quantum circuit Q using the oracles $\{U, U^\dagger, A_\phi, B_\phi\}$ such that

$$|\langle A_0 | Q | B_0 \rangle| \rightarrow 1,$$

under the assumption that the original matrix element $\langle A_0 | U | B_0 \rangle$ is nonzero. We now show that this problem can be solved without prior knowledge of the specific value of $\langle A_0 | U | B_0 \rangle$. The solution follows directly from theorem 3.1.1. The key insight lies in recognizing that the problem naturally gives rise to two concentric Bloch spheres: one associated with the action of U on the privileged subspace defined by $|B_0\rangle$, and the other associated with measurements in the $|A_0\rangle$ basis (Figure 3.1).

Specifically, the quantum state $U|B_0\rangle$ has a non-zero component along $|A_0\rangle$ and another component orthogonal to it. We define:

$$|A_\perp\rangle = \alpha (I - |A_0\rangle\langle A_0|) U |B_0\rangle,$$

where α is the normalization factor that ensures $|A_\perp\rangle$ is a unit vector. Then,

$$U |B_0\rangle = a |A_0\rangle + \sqrt{1 - a^2} |A_\perp\rangle,$$

with $a = \langle A_0 | U | B_0 \rangle$. Moreover, we define a state $|B_\perp\rangle$ such that:

$$U |B_\perp\rangle = -a |A_\perp\rangle + \sqrt{1 - a^2} |A_0\rangle.$$

Therefore, the action of U on the two-dimensional Hilbert space spanned by

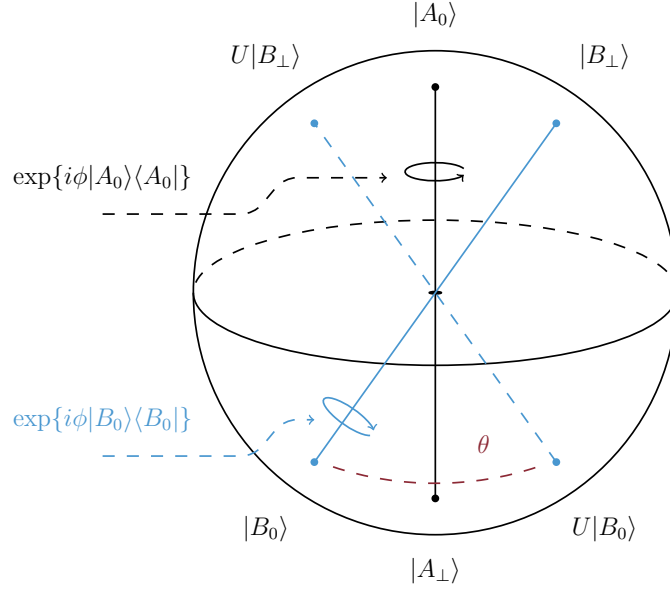


Figure 3.1: Illustration of amplitude amplification on the Bloch sphere: starting from $|B_0\rangle$, controlled rotations about $|B_0\rangle$ (blue) and $|A_0\rangle$ (black) drive the state toward the target $|A_0\rangle$ (north pole).

$\{|B_0\rangle, |B_\perp\rangle\}$ and $\{|A_0\rangle, |A_\perp\rangle\}$ can be represented in matrix form as:

$$U = R(a) = \begin{matrix} & |B_0\rangle & |B_\perp\rangle \\ \begin{matrix} |A_0\rangle \\ |A_\perp\rangle \end{matrix} & \begin{bmatrix} a & \sqrt{1-a^2} \\ \sqrt{1-a^2} & -a \end{bmatrix} \end{matrix}, \quad (3.9)$$

It is now clear how U maps a state in the $\{|B_0\rangle, |B_\perp\rangle\}$ basis to a state in the $\{|A_0\rangle, |A_\perp\rangle\}$ basis.

Finally, we can state the following theorem:

Theorem 3.2.1. Given a unitary U , its inverse U^\dagger , and phase operators defined as

$$A_\phi = e^{i\phi|A_0\rangle\langle A_0|}, \quad B_\phi = e^{i\phi|B_0\rangle\langle B_0|},$$

we have:

$$\langle A_0 | \left[\prod_{k=1}^{\frac{d}{2}} U B_{\phi_{2k-1}} U^\dagger A_{\phi_{2k}} \right] U | B_0 \rangle = P(a), \quad (3.10)$$

where $P(a)$ is a degree- d polynomial in $a = \langle A_0 | U | B_0 \rangle$ that satisfies the constraints of Theorem [3.1.1](#)

Proof. We begin by rewriting eq. (3.10) in the basis we are working with, in order to express it in a form analogous to eq. (3.5). First, we observe that:

$$U^\dagger = \begin{array}{c} |B_0\rangle \\ |B_\perp\rangle \end{array} \begin{array}{cc} \begin{array}{c} |A_0\rangle \\ |A_\perp\rangle \end{array} \\ \begin{bmatrix} a & \sqrt{1-a^2} \\ \sqrt{1-a^2} & -a \end{bmatrix} \end{array}. \quad (3.11)$$

We then define the operators Z_A and Z_B as the Pauli- Z matrices expressed in the $\{|A_0\rangle, |A_\perp\rangle\}$ and $\{|B_0\rangle, |B_\perp\rangle\}$ bases, respectively. Correspondingly, we define the Z -rotations as:

$$A_\phi = e^{i\phi Z_A}, \quad B_\phi = e^{i\phi Z_B}.$$

These rotations act as phase shifts around the Z -axis in each respective basis. Note that these definitions are equivalent to the operators in eq. (3.7), up to a global phase, which is physically irrelevant and can be safely ignored. Therefore, there is no ambiguity or inconsistency in the notation when referring to A_ϕ and B_ϕ in either form.

We now express the matrix $W(a)$ from eq. (3.1) in terms of our chosen basis:

$$U = -i e^{i\frac{\pi}{4} Z_A} W(a) e^{i\frac{\pi}{4} Z_B}, \quad (3.12)$$

and similarly, its adjoint is given by:

$$U^\dagger = i e^{-i\frac{\pi}{4} Z_B} W(a) e^{-i\frac{\pi}{4} Z_A}. \quad (3.13)$$

We use the same notation $W(a)$ in both cases, as the basis in use is clear from the context.

Finally, we rewrite eq. (3.10) as:

$$\langle A_0 | \left[\prod_{k=1}^{\frac{d}{2}} \left(-i e^{i\frac{\pi}{4} Z_A} W(a) e^{i\frac{\pi}{4} Z_B} \right) B_{\phi_{2k-1}} \left(-i e^{i\frac{\pi}{4} Z_B} W(a) e^{i\frac{\pi}{4} Z_A} \right) A_{\phi_{2k}} \right] U | B_0 \rangle, \quad (3.14)$$

which can be rewritten in the compact form:

$$\langle A_0 | \left(e^{i\phi'_0 Z} \prod_{k=1}^d W(a) e^{i\phi'_k Z} \right) U | B_0 \rangle, \quad (3.15)$$

where the modified phases $\{\phi'_k\}$ are linear combinations of the original phases $\{\phi_k\}$. \square

3.3 Quantum eigenvalue transform

As demonstrated in Theorem [3.2.1](#), we can apply a polynomial transformation to a matrix element, namely $\langle A_0 | U | B_0 \rangle$. This technique extends to transformations over an entire vector space. In particular, we show that the same algorithm can be used to transform all eigenvalues of an hermitian matrix \mathcal{H} , once it has been embedded into a unitary matrix U . To perform such an embedding, we introduce an ancilla qubit, an auxiliary qubit used to control the application of quantum operations. We suppose we have the unitary U :

$$U = \begin{matrix} & \begin{matrix} 0 & 1 \end{matrix} \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{bmatrix} \mathcal{H} & \cdot \\ \cdot & \cdot \end{bmatrix} \end{matrix}. \quad (3.16)$$

We note that the indices adjacent to the matrix representation of U indicate how the Hamiltonian \mathcal{H} is encoded. Specifically, we have $\mathcal{H} = \langle 0 | U | 0 \rangle$. Since \mathcal{H} is encoded in a unitary matrix, we have the constraint that $\|\mathcal{H}\| \leq 1$. This may seem like a strong limitation, but we can always scale the Hamiltonian by its norm and encode $\frac{\mathcal{H}}{\|\mathcal{H}\|}$ instead of \mathcal{H} . The Hamiltonian operator \mathcal{H} can be decomposed into its eigenspace as:

$$\mathcal{H} = \sum_{\lambda} \lambda |\lambda\rangle \langle \lambda|. \quad (3.17)$$

We now proceed by focusing on a specific unitary U for clarity. In particular, we complete the missing block as:

$$U = \begin{bmatrix} \mathcal{H} & \sqrt{I - \mathcal{H}^2} \\ \sqrt{I - \mathcal{H}^2} & -\mathcal{H} \end{bmatrix}, \quad (3.18)$$

where:

$$\sqrt{I - \mathcal{H}^2} = \sum_{\lambda} \sqrt{1 - \lambda^2} |\lambda\rangle \langle \lambda| .$$

However, a general U does not need to take this specific form, although even a general one takes a structure similar to eq. (3.18) [9]. Thus, we assume that U takes this form; in particular, it can be expressed as a sum of two tensor products:

$$U = Z \otimes \mathcal{H} + X \otimes \sqrt{I - \mathcal{H}^2} ,$$

and it acts as:

$$\begin{aligned} U |0\rangle |\lambda\rangle &= \lambda |0\rangle |\lambda\rangle + \sqrt{1 - \lambda^2} |1\rangle |\lambda\rangle , \\ U |1\rangle |\lambda\rangle &= \sqrt{1 - \lambda^2} |0\rangle |\lambda\rangle - \lambda |1\rangle |\lambda\rangle , \end{aligned}$$

This indicates that U acts on a qubit basis (i.e., $\{|0\rangle |\lambda\rangle, |1\rangle |\lambda\rangle\}$) for each eigenvalue of \mathcal{H} . Consequently, U can be expressed as a direct sum over separate Bloch spheres:

$$\begin{aligned} U &= \bigoplus_{\lambda} \begin{bmatrix} \lambda & \sqrt{1 - \lambda^2} \\ \sqrt{1 - \lambda^2} & -\lambda \end{bmatrix} \otimes |\lambda\rangle \langle \lambda| \\ &= \bigoplus_{\lambda} R(\lambda) \otimes |\lambda\rangle \langle \lambda| ; \end{aligned} \tag{3.19}$$

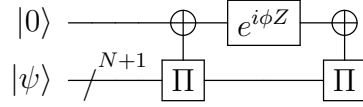
Here, $R(\lambda)$ is defined as in eq. (3.9). In this way, we can reformulate the problem in a manner reminiscent of amplitude amplification. However, instead of operating within two concentric Bloch spheres, we are dealing with N independent two-dimensional subspaces—each associated with a different eigenvalue—that evolve in parallel. In each subspace, the evolution corresponds to a phase rotation induced by the operator U . Nevertheless, there are still distinct vector spaces in which the input and the output of \mathcal{H} exist. This vector space is defined by the projector $\Pi = |0\rangle \langle 0|$ acting on the ancilla qubit. Generalizing the way in which amplitude amplification uses the phase shift A_{ϕ} acting on a one dimensional vector space $|A_0\rangle \langle A_0|$, we may now define a projector-controlled phase-shift operation Π_{ϕ} :

$$\Pi_{\phi} := e^{i2\phi\Pi} ,$$

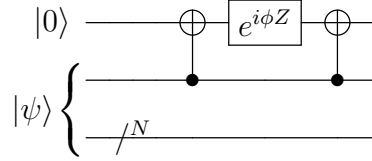
which gives a phase of $e^{i2\phi}$ to the entire subspace determined by Π . Note that we may define this operator as eq. (3.7):

$$\Pi_\phi := e^{i\phi Z}. \quad (3.20)$$

These two definitions are equivalent since they differ only by a global phase that may be neglected. The Π_ϕ operator can be implemented in a quantum circuit by employing two instances of projector-controlled-NOT-gates around a single-qubit z -rotation by angle ϕ on one extra ancilla qubit:



In particular, in the simple case of eq. (3.16), the projector-controlled-NOT gate reduces to a standard CNOT gate. Therefore, the circuit becomes:



It is important to observe that on each eigenspace of eq. (3.19), Π_ϕ acts as a z -rotation:

$$\Pi_\phi = \bigoplus_{\lambda} e^{i\phi Z} \otimes |\lambda\rangle \langle \lambda|.$$

Finally, having established the intuitive basis, we now state the following theorem.

Theorem 3.3.1. Given a block encoding of Hamiltonian $\mathcal{H} = \sum_{\lambda} \lambda |\lambda\rangle \langle \lambda|$ in a unitary matrix U :

$$U = \begin{matrix} & \Pi \\ \Pi & \begin{bmatrix} \mathcal{H} & \cdot \\ \cdot & \cdot \end{bmatrix} \end{matrix},$$

with the location of \mathcal{H} determined by projector Π and given the ability to perform Π -controlled-NOT operations to realize projection-controlled phase-

shift operations Π_ϕ , then, for even d :

$$U_{\vec{\phi}} = \left[\prod_{k=1}^{\frac{d}{2}} \Pi_{\phi_{2k-1}} U^\dagger \Pi_{\phi_{2k}} U \right] = {}^\Pi \left[\begin{array}{cc} \text{Poly}(\mathcal{H}) & \cdot \\ \cdot & \cdot \end{array} \right],$$

where

$$\text{Poly}(\mathcal{H}) = \sum_{\lambda} \text{Poly}(\lambda) |\lambda\rangle \langle \lambda| ,$$

is a polynomial transform of the eigenvalues of \mathcal{H} . The polynomial is of degree at most d and obeys the conditions on P from Theorem 3.1.1. Similarly, for d odd:

$$U_{\vec{\phi}} = \Pi_{\phi_1} U \left[\prod_{k=1}^{\frac{d-1}{2}} \Pi_{\phi_{2k}} U^\dagger \Pi_{\phi_{2k+1}} U \right] = {}^\Pi \left[\begin{array}{cc} \text{Poly}(\mathcal{H}) & \cdot \\ \cdot & \cdot \end{array} \right],$$

where $\text{Poly}(\mathcal{H})$ has an analogous interpretation.

The idea of the proof is to use eq. (3.19) and apply Theorem 3.2.1 independently to each Bloch sphere. In this way, we obtain:

$$U_{\vec{\phi}} = \bigoplus_{\lambda} \left[\prod_{k=1}^{\frac{d}{2}} e^{i\phi_{2k-1}Z} R(\lambda) e^{i\phi_{2k}Z} R(\lambda) \right] \otimes |\lambda\rangle \langle \lambda| , \quad (3.21)$$

where we used the fact that $R^\dagger(\lambda) = R(\lambda)$ in eq. (3.19), as in eq. (3.11). Finally, we can apply Theorem 3.2.1 to each Bloch sphere component independently.

This algorithm can be generalized to matrices that are not hermitian by using their singular value decomposition (SVD) instead of the eigenvalue decomposition:

$$A = W \Sigma V^\dagger = \sum_k \sigma_k |w_k\rangle \langle v_k| , \quad (3.22)$$

Using this decomposition, we can apply the same idea introduced in eq. (3.17) to perform polynomial transformations on the singular values. This generalization allows us to apply these transformations to arbitrary linear operators.

Within this broader framework, we can unify and generalize a wide range of quantum algorithms by viewing them as instances of polynomial transformations on the spectrum of an operator embedded in a unitary matrix. This includes *quantum search algorithms*, such as Grover’s algorithm, which amplifies the amplitude of a marked state; *eigenvalue thresholding*, where the goal is to project onto a subspace defined by eigenvalues above or below a certain threshold; *quantum phase estimation*, which extracts eigenvalue information with exponential precision; and *function evaluation problems*, where the goal is to compute a function of a Hermitian operator, such as the sign, exponential, or inverse. These seemingly different problems can be tackled using a unified polynomial-based approach. For an in-depth treatment of these applications and the underlying framework, we refer the reader to [\[4\]](#).

Chapter 4

Block encodings

In this chapter, we introduce the concept of *block encoding* for matrices and examine the key features of the quantum circuits used to implement it. Our aim is to develop a clear and practical understanding of how such circuits can be constructed, combined, and manipulated.

The central goal is to study quantum circuits U_A that encode a matrix A in such a way that, when applied to an input state $|\psi\rangle$, they produce an output state proportional to $A|\psi\rangle$. This capability is fundamental for realizing more advanced quantum algorithms based on the Quantum Singular Value Transformation (QSVT) framework introduced in the previous chapter.

4.1 General notion

Block-encoding is a technique for embedding a properly scaled nonunitary matrix $A \in \mathbb{C}^{N \times N}$, with $N = 2^n$ (i.e a n -qubit matrix) into an $n + 1$ -qubit unitary matrix U_A of the form:

$$U_A = \begin{matrix} & \begin{matrix} 0 & 1 \end{matrix} \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{bmatrix} A & * \\ * & * \end{bmatrix} \end{matrix}. \quad (4.1)$$

where the symbol $*$ denotes matrix blocks whose values are not yet specified. This structure is useful since if we want to apply A to a state $|\psi\rangle$ we can

apply the unitary matrix U_A to the state $|0\rangle |\psi\rangle$ and get:

$$U_A |0\rangle |\psi\rangle = \begin{bmatrix} A\psi \\ * \end{bmatrix}$$

$$|0\rangle |\psi\rangle \xrightarrow{U_A} |0\rangle (A|\psi\rangle) + |\perp\rangle. \quad (4.2)$$

Here, the (unnormalized) state $|\perp\rangle$ can be written as $|1\rangle |\xi\rangle$ for some (unnormalized) state $|\xi\rangle$, which is irrelevant to the computation of $A|\psi\rangle$. In particular, it satisfies the following orthogonality relation:

$$(\langle 0| \otimes I_n) |\perp\rangle = 0. \quad (4.3)$$

Then, if we measure the state $|0\rangle$ on the first register and get the outcome 0, we will get $A|\psi\rangle$. This process can be summarized by the following quantum circuit:

$$\begin{array}{c} |0\rangle \\ |\psi\rangle \end{array} \xrightarrow{U_A} \begin{array}{c} \boxed{\text{Measurement}} \\ \text{---} \end{array} = \begin{array}{c} 0 \\ \frac{A|\psi\rangle}{\|A|\psi\rangle\|} \end{array} \quad (4.4)$$

Note that the output state is normalized after the measurement takes place, and note that the probability of success is $\|A|\psi\rangle\|^2$. It is important to remark that a necessary condition for the existence of the unitary operator U_A is that $\|A\| \leq 1$. However, the constraint on the norm of A is not a problem, indeed it is possible to find a sufficiently large scaling factor α such that we encode $\hat{A} = \frac{1}{\alpha}A$. Nevertheless the factor α will lower the probability of success, as it holds that:

$$p = \frac{\|A|\psi\rangle\|^2}{\alpha^2}.$$

So if α is chosen to be too large, the probability of obtaining $|0\rangle$ from the measurement can be vanishingly small. We may not need to restrict the matrix U_A to be an $(n+1)$ -qubit matrix. If we can find any $(n+m)$ -qubit matrix U_A such that :

$$U_A = \begin{bmatrix} \frac{1}{\alpha}A & * & \dots & * \\ * & * & \dots & * \\ \vdots & & \ddots & \\ * & \dots & & * \end{bmatrix},$$

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

$$\frac{1}{\alpha} A = (\langle 0^m | \otimes I_n) U_A (| 0^m \rangle \otimes I_n). \quad (4.5)$$

Moreover, it can be difficult to find U_A to block encode A exactly, for this reason it may be sufficient to construct U_A to block encode A up to some error ϵ . Finally we give the formal definition of block encoding.

Definition 4.1.1 (Def 43 [3]). Let $a, n, m \in \mathbb{N}, m = a + n$. Then an m -qubit unitary U is a (α, a, ϵ) -block encoding of an n -qubit linear operator A if

$$\|\alpha (\langle 0^{\otimes a} | \otimes I_n) U (| 0^{\otimes a} \rangle \otimes I_n) - A\| \leq \epsilon. \quad (4.6)$$

In this case, the unitary operator U_A is referred to as an (α, m, ϵ) -block-encoding of the matrix A . When $\epsilon = 0$ the block-encoding is said to be exact and the unitary matrix is called an (α, m) -block-encoding of A . We denote the set of all (α, m, ϵ) -block-encodings of A by $\text{BE}_{\alpha, m}(A, \epsilon)$, and define $\text{BE}_{\alpha, m}(A) = \text{BE}_{\alpha, m}(A, 0)$ as the set of all exact (α, m) -block-encodings of A .

Remark 4.1.1. The following statements are equivalent:

1. U is an (α, k) -block-encoding of A ;
2. For every n -qubit state $|\psi\rangle$, we have

$$U | 0^{\otimes k} \rangle |\psi\rangle = \frac{1}{\alpha} | 0^{\otimes k} \rangle A |\psi\rangle + |\perp\rangle ;$$

3. For every $0 \leq i, j \leq N - 1$ we have

$$(\langle 0^{\otimes k} | \otimes \langle i |) U (| 0^{\otimes k} \rangle \otimes | j \rangle) = \frac{a_{ij}}{\alpha}.$$

The following example is called unitary dilation of A and it is a generalization of block-encoding shown in eq. (3.18):

Example 4.1.1. For any n -qubit matrix A with $\|A\| \leq 1$, the singular value decomposition (SVD) of A is denoted by $A = U\Sigma V^\dagger$, 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 as follows:

$$\begin{aligned} U_A &:= \begin{bmatrix} U & 0 \\ 0 & I_n \end{bmatrix} \begin{bmatrix} \Sigma & \sqrt{I_n - \Sigma^2} \\ \sqrt{I_n - \Sigma^2} & -\Sigma \end{bmatrix} \begin{bmatrix} V^\dagger & 0 \\ 0 & I_n \end{bmatrix} \\ &= \begin{bmatrix} A & U \cdot \sqrt{I_n - \Sigma^2} \\ \sqrt{I_n - \Sigma^2} \cdot V^\dagger & -\Sigma \end{bmatrix} \end{aligned} \quad (4.7)$$

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

However, this method is not practical in general, as it requires explicit knowledge of the singular value decomposition of A . Furthermore, it does not provide any guidance on how to implement the resulting unitary U_A efficiently as a quantum circuit.

We are now interested in studying how we can compute the operations between block-encoding matrices. The first two lemmas can be found in [3]. We start with linear combination of block-encoded matrices. To formalize the subsequent result, we need the following definition.

Definition 4.1.2 (State preparation pair). Let $y \in \mathbb{C}^m$ with $\|y\|_1 \leq \beta$. A pair of unitaries matrices (P_L, P_R) is called a (β, b, ϵ) -state preparation pair if the following conditions hold:

1. $P_L |0^{\otimes b}\rangle = \sum_{j=0}^{2^b-1} c_j |j\rangle$,
2. $P_R |0^{\otimes b}\rangle = \sum_{j=0}^{2^b-1} d_j |j\rangle$,

such that

$$\sum_{j=0}^{m-1} |\beta c_j^* d_j - y_j| \leq \epsilon, \quad (4.8)$$

and for all $j \in \{m, \dots, 2^b - 1\}$, it holds that $c_j^* d_j = 0$.

This definition closely resembles that of block-encoding; however, there are important differences: here we are working with vectors rather than

matrices, we aim to construct two quantum states (via state preparation unitaries) instead of a single quantum gate, and the relevant quantity is the 1-norm rather than the operator norm.

In the next proposition, we will use it to derive a block-encoding construction.

Lemma 4.1.1 ([3]). Let $A = \sum_{j=1}^m y_j A_j$ be an s -qubit operator, and let $\epsilon \in \mathbb{R}^+$. Assume that (P_L, P_R) is a (β, b, ϵ_1) -state-preparation-pair for the vector y , and let

$$W = \sum_{j=0}^{m-1} |j\rangle \langle j| \otimes U_j + \left(I - \sum_{j=0}^{m-1} |j\rangle \langle j| \right) \otimes I_a \otimes I_s$$

be an $(s + a + b)$ -qubit unitary operator such that for all $j \in \{0, \dots, m-1\}$, the operator U_j is an (α, a, ϵ_2) -block-encoding of A_j .

Then, it is possible to implement an $(\alpha\beta, a+b, \alpha\epsilon_1 + \alpha\beta\epsilon_2)$ -block-encoding of A using a single application 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 a $(\alpha\beta, a+b, \alpha\epsilon_1 + \alpha\beta\epsilon_2)$ -block-encoding of A :

$$\begin{aligned} & \left\| A - \alpha\beta (\langle 0^{\otimes b} | \otimes \langle 0^{\otimes a} | \otimes I) \tilde{W} (|0^{\otimes b}\rangle \otimes |0^{\otimes a}\rangle \otimes I) \right\| \\ &= \left\| A - \alpha\beta \left(\sum_{k=0}^{2^b-1} c_k^* \langle k | \otimes \langle 0^{\otimes a} | \otimes I \right) \left(\sum_{j=0}^{m-1} |j\rangle \langle j| \otimes U_j + \left(I - \sum_{j=0}^{m-1} |j\rangle \langle j| \right) \otimes I_a \otimes I_s \right) \right. \\ & \quad \left. \cdot \left(\sum_{i=0}^{2^b-1} d_i |i\rangle \otimes |0^{\otimes a}\rangle \otimes I \right) \right\| \\ &= \left\| A - \alpha \left(\sum_{j=0}^{m-1} \beta(c_j^* d_j) (\langle 0^{\otimes a} | \otimes I) U_j (|0^{\otimes a}\rangle \otimes I) \right) + \right. \\ & \quad \left. + \alpha\beta \left(\sum_{j=m}^{2^b-1} (c_j^* d_j) (\langle 0^{\otimes a} | \otimes I) (|0^{\otimes a}\rangle \otimes I) \right) \right\| \end{aligned}$$

The second term will be the sum of zeros since we have by definition

$c_j^* d_j = 0$ for all $j \in \{m, \dots, 2^b - 1\}$. Then we have:

$$\left\| A - \alpha \left(\sum_{j=0}^{m-1} \beta(c_j^* d_j) (\langle 0^{\otimes a} | \otimes I) U_j (|0^{\otimes a}\rangle \otimes I) \right) \right\|$$

We can add and subtract the term $\sum_{j=0}^{m-1} y_j (\langle 0^{\otimes a} | \otimes I) U_j (|0^{\otimes a}\rangle \otimes I)$ and, using the triangle inequality and the property that $\sum_{j=0}^{m-1} |\beta c_j^* d_j - y_j| \leq \epsilon_1$ we get :

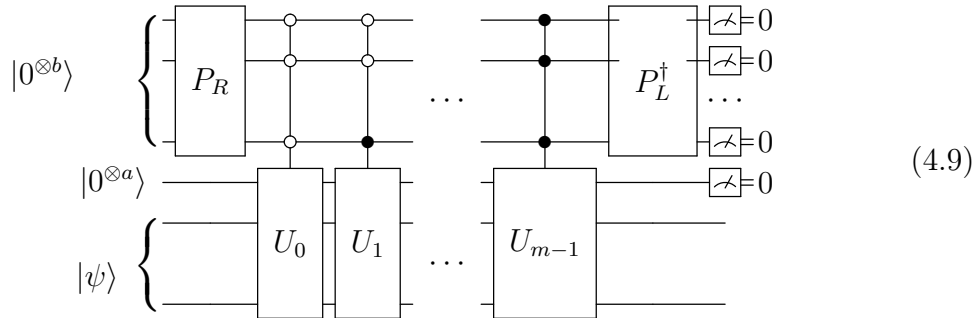
$$\leq \alpha \epsilon_1 + \left\| A - \alpha \left(\sum_{j=0}^{m-1} y_j (\langle 0^{\otimes a} | \otimes I) U_j (|0^{\otimes a}\rangle \otimes I) \right) \right\|$$

Now we use the triangle inequality again with the fact that we can write $A = \sum_{j=0}^{m-1} y_j A_j$ and we get:

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

□

The circuit will have the following general form :



Remark 4.1.2. One of the most widely used techniques for constructing block encodings is based on the Linear Combination of Unitaries (LCU) method, which expresses a target matrix as a weighted sum of unitary matrices. A common and practical choice is to use tensor products of Pauli matrices, not only because they are inexpensive to implement on quantum

hardware, but also because many Hamiltonians are already naturally written in this form. This avoids the need to explicitly compute the decomposition. Moreover, this approach does not require additional ancilla qubits beyond those used to encode the coefficients. However, the main drawback of LCU is that it often requires implementing multi-controlled unitary operations, whose decomposition into elementary gates is resource-intensive and scales poorly with the number of controls.

The following operation is the product of block-encoded matrices. 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 4.1.2 ([3]). Let U be an (α, a, δ) -block-encoding of an n -qubit operator A , V be an (β, b, ϵ) -block-encoding of an n -qubit operator B , and S the SWAP operation between the ancilla qubits:

$$S|\psi\rangle|\xi\rangle = |\xi\rangle|\psi\rangle, \quad \forall |\psi\rangle \in \mathbb{C}^{2^a}, |\xi\rangle \in \mathbb{C}^{2^b} \quad (4.10)$$

then $(I_b \otimes U)(S \otimes I_n)(I_a \otimes V)$ is a $(\alpha\beta, a+b, \alpha\epsilon + \beta\delta)$ -block-encoding of AB .

Proof.

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

Another way to prove is to check whether the circuit will output $|0^{\otimes b}\rangle |0^{\otimes a}\rangle \frac{\tilde{A}}{\alpha} \frac{\tilde{B}}{\beta} |\psi\rangle$

$$\begin{aligned}
& (I_b \otimes U)(S \otimes I_n)(I_a \otimes V)(|0^{\otimes a+b}\rangle |\psi\rangle) \\
&= (I_b \otimes U)(S \otimes I_n)(|0^{\otimes a}\rangle (|0^{\otimes b}\rangle \frac{\tilde{B}}{\beta} |\psi\rangle + |\perp\rangle)) \\
&= (I_b \otimes U)(|0^{\otimes b}\rangle |0^{\otimes a}\rangle \frac{\tilde{B}}{\beta} |\psi\rangle + |\perp\rangle) \\
&= |0^{\otimes b}\rangle |0^{\otimes a}\rangle \frac{\tilde{A}}{\alpha} \frac{\tilde{B}}{\beta} |\psi\rangle + |\perp\rangle + |\perp_2\rangle
\end{aligned}$$

Then our final operator will be $\frac{\tilde{A}}{\alpha} \frac{\tilde{B}}{\beta}$ with an error $\alpha\epsilon + \beta\delta$ \square

The circuit will have the following general form:

$$\begin{array}{c}
|0^{\otimes b}\rangle \xrightarrow{/b} \text{---} \times \xrightarrow{/a} \text{---} \boxed{\text{---}} = 0 \\
|0^{\otimes a}\rangle \xrightarrow{/a} \text{---} \boxed{U} \text{---} \times \xrightarrow{/b} \text{---} \boxed{V} \text{---} \boxed{\text{---}} = 0 \\
|\psi\rangle \xrightarrow{/n} \text{---} \boxed{U} \text{---} \boxed{V} \text{---} \text{---}
\end{array} \quad (4.11)$$

The following operation creates a block-encoding of a Kronecker product of two matrices from the block-encodings of the individual matrices [18].

Lemma 4.1.3. Let U_n and U_m be (α, a, ϵ_1) - and (β, b, ϵ_2) -block-encodings of A_s and A_t , respectively, and S defined as in eq. (4.10). Then,

$$S(U_n \otimes U_m)S$$

is an $(\alpha\beta, a+b, \alpha\epsilon_2 + \beta\epsilon_1 + \epsilon_1\epsilon_2)$ -block-encoding of $A_s \otimes A_t$.

Proof.

$$\begin{aligned}
& \|A_s \otimes A_t - \alpha\beta(\langle 0^{\otimes a+b}| \otimes I_s \otimes I_t)S_{n+m}(U_n \otimes U_m)S_{n+m}^\dagger(|0^{\otimes a+b}\rangle \otimes I_s \otimes I_t)\| \\
&= \|A_s \otimes A_t - \alpha\beta(\langle 0^{\otimes a}| \otimes I_s \otimes \langle 0^{\otimes b}| \otimes I_t)(U_n \otimes U_m)(|0^{\otimes a}\rangle \otimes I_s \otimes |0^{\otimes b}\rangle \otimes I_t)\| \\
&= \|A_s \otimes A_t - \overbrace{\alpha(\langle 0^{\otimes a}| \otimes I_s)U_n(|0^{\otimes a}\rangle \otimes I_s)}^{\tilde{A}_s} \otimes \overbrace{\beta(\langle 0^{\otimes b}| \otimes I_t)U_m(|0^{\otimes b}\rangle \otimes I_t)}^{\tilde{A}_t}\| \\
&= \|A_s \otimes A_t - \tilde{A}_s \otimes \tilde{A}_t\| \\
&\leq \|(\tilde{A}_s + \epsilon_1 I) \otimes (\tilde{A}_t + \epsilon_2 I) - \tilde{A}_s \otimes \tilde{A}_t\| \\
&= \|\tilde{A}_s \otimes \epsilon_2 I_t + \epsilon_1 I_s \otimes \tilde{A}_t + \epsilon_1 I_s \otimes \epsilon_2 I_t\| \\
&\leq \epsilon_2 \|\tilde{A}_s\| + \epsilon_1 \|\tilde{A}_t\| + \epsilon_1 \epsilon_2 \\
&\leq \alpha\epsilon_2 + \beta\epsilon_1 + \epsilon_1 \epsilon_2
\end{aligned}$$

where we used that $\|\tilde{A}\| \leq \alpha$ and analogous result for \tilde{A}_t . \square

The circuit will have the following general form:

$$(4.12)$$

These operations can also be combined with each other, allowing us to block-encode matrices with more sophisticated structures.

4.2 Fast Approximate Quantum Circuits for Block-Encodings

The following section closely follows the treatment presented in [10].

Definition 4.2.1. The matrix query operation \mathcal{O}_A of a matrix A applies:

$$\mathcal{O}_A |0\rangle |i\rangle |j\rangle \rightarrow \left(\hat{a}_{ij} |0\rangle + \sqrt{1 - |\hat{a}_{ij}|^2} |1\rangle \right) |i\rangle |j\rangle$$

with $\hat{a}_{ij} = \frac{a_{ij}}{\|A\|_\infty}$.

The following theorem shows how to construct a block-encoding with in terms of a matrix query oracle \mathcal{O}_A

Theorem 4.2.1. The circuit in eq. (4.13) is an $(\frac{1}{2^n \cdot \|A\|_\infty}, n+1)$ -block-encoding of A .

$$(4.13)$$

Proof. The circuit U_A can be written in matrix notation as

$$U_A = (I_1 \otimes H^{\otimes n} \otimes I_n)(I_1 \otimes S)\mathcal{O}_A(I_1 \otimes H^{\otimes n} \otimes I_n).$$

To prove that U_A is an $(\frac{1}{2^n \cdot \|A\|_\infty}, n+1)$ -block-encoding of A we just need to verify that

$$\langle 0| \langle 0^{\otimes n}| \langle i| U_A |0\rangle |0^{\otimes n}\rangle |j\rangle = \frac{1}{2^n \cdot \|A\|_\infty} a_{ij}$$

On the one hand, we have

$$\begin{aligned}
|0\rangle |0^{\otimes n}\rangle |j\rangle &\xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |0\rangle |k\rangle |j\rangle \\
&\xrightarrow{\mathcal{O}_A} \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \left(\hat{a}_{kj} |0\rangle + \sqrt{1 - |\hat{a}_{kj}|^2} |1\rangle \right) |k\rangle |j\rangle \\
&\xrightarrow{S} \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \left(\hat{a}_{kj} |0\rangle + \sqrt{1 - |\hat{a}_{kj}|^2} |1\rangle \right) |j\rangle |k\rangle,
\end{aligned}$$

while, on the other hand, we get

$$|0\rangle |0^{\otimes n}\rangle |i\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} |0\rangle |l\rangle |i\rangle.$$

Combining both yields :

$$\begin{aligned}
\langle 0| \langle 0^{\otimes n}| \langle i| U_A |0\rangle |0^{\otimes n}\rangle |j\rangle &= \frac{1}{2^n} \left(\sum_{l=0}^{2^n-1} \langle 0| \langle l| \langle i| \right) \left(\sum_{k=0}^{2^n-1} \left(\hat{a}_{kj} |0\rangle + \sqrt{1 - |\hat{a}_{kj}|^2} |1\rangle \right) |j\rangle |k\rangle \right) \\
&= \frac{1}{2^n} \sum_{k=0}^{2^n-1} \sum_{l=0}^{2^n-1} \hat{a}_{kj} \langle l|j\rangle \langle i|k\rangle \\
&= \frac{1}{2^n} \hat{a}_{ij} = \frac{1}{2^n \cdot \|A\|_\infty} a_{ij}.
\end{aligned}$$

□

We present how the oracle \mathcal{O}_A can be implemented in simple 1- and 2-qubit gates for arbitrary matrices [10]. We first consider real-valued matrix. In this case, for given row and column indices i and j , \mathcal{O}_A acts on the $|0\rangle$ state of the first qubit as an R_y gate with angle

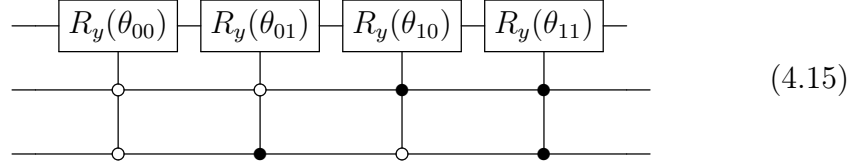
$$\theta_{ij} = 2 \arccos(a_{ij}) \quad (4.14)$$

i.e.,

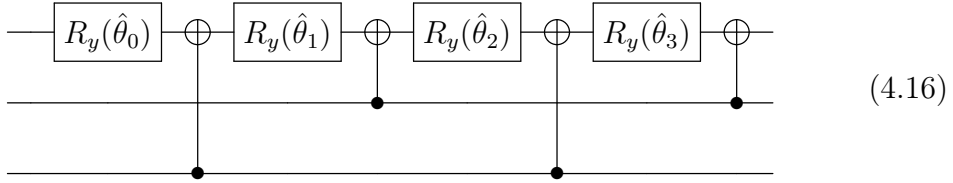
$$R_y(\theta_{ij}) = \begin{bmatrix} \cos(\frac{\theta_{ij}}{2}) & -\sin(\frac{\theta_{ij}}{2}) \\ \sin(\frac{\theta_{ij}}{2}) & \cos(\frac{\theta_{ij}}{2}) \end{bmatrix}$$

A first naive implementation of the oracle uses N^2 multi-controlled R_y gates. We will denote by $C^n(R_y)$ an R_y gate with n control qubits. The circuit

construction for \mathcal{O}_A uses one $C^n(R_y)$ gate for each matrix entry a_{ij} where the control qubits encode the row and column indices $|i\rangle|j\rangle$ of the corresponding entry. The circuit illustrated below represent the oracle \mathcal{O}_A for the encoding of a 2×2 real matrix using 3 qubits and 4 $C^2(R_y)$ gates.



The major disadvantage of this naive approach is that it requires $N^2 C^{2n}(R_y)$ gates to implement the \mathcal{O}_A oracle for $A \in \mathbb{R}^{N \times N}$. However, every $C^{2n}(R_y)$ requires $O(N^2)$ 1- and 2-qubits gates to be implemented [19]. This brings the total gate complexity of this circuit to $O(N^4)$ which is excessive as it is quadratically worse than the classical representation cost. Then we can use the FABLE (Fast Approximate Quantum Circuits for Block-Encodings) implementation [10] to have a quadratic reduction in gate complexity. We can rewrite the circuit with another structure as derived in [20]. Let us first see before the construction for a small-scale example of $A \in \mathbb{R}^{2 \times 2}$.



In eq. (4.16), the angles $\hat{\theta}_0, \hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3$ are computed from the data $A \in \mathbb{R}^{2 \times 2}$ as we will explain later. We analyze the action of the above circuit based on the following two elementary properties of R_y rotations:

$$\begin{aligned} R_y(\alpha)R_y(\beta) &= R_y(\alpha + \beta) \\ X R_y(\alpha) X &= R_y(-\alpha) \end{aligned}$$

Then it follows that the first qubit is rotated as

$$\begin{aligned}
00 : \quad & R_y(\hat{\theta}_3) \cdot R_y(\hat{\theta}_2) \cdot R_y(\hat{\theta}_1) \cdot R_y(\hat{\theta}_0) = R_y(\hat{\theta}_3 + \hat{\theta}_2 + \hat{\theta}_1 + \hat{\theta}_0) \\
01 : \quad & R_y(\hat{\theta}_3) \cdot X \cdot R_y(\hat{\theta}_2) \cdot R_y(\hat{\theta}_1) \cdot X \cdot R_y(\hat{\theta}_0) = R_y(\hat{\theta}_3 - \hat{\theta}_2 - \hat{\theta}_1 + \hat{\theta}_0) \\
10 : \quad & X \cdot R_y(\hat{\theta}_3) \cdot R_y(\hat{\theta}_2) \cdot X \cdot R_y(\hat{\theta}_1) \cdot R_y(\hat{\theta}_0) = R_y(-\hat{\theta}_3 - \hat{\theta}_2 + \hat{\theta}_1 + \hat{\theta}_0) \\
11 : \quad & X \cdot R_y(\hat{\theta}_3) \cdot X \cdot R_y(\hat{\theta}_2) \cdot X \cdot R_y(\hat{\theta}_1) \cdot X \cdot R_y(\hat{\theta}_0) = R_y(-\hat{\theta}_3 + \hat{\theta}_2 - \hat{\theta}_1 + \hat{\theta}_0)
\end{aligned}$$

where the rotation angle depends on the state of the control qubits as indicated above. To implement an \mathcal{O}_A oracle with angles $\theta_{00}, \theta_{01}, \theta_{10}, \theta_{11}$, as given by [4.14](#), we vectorize A in row-major such that $\text{vec}(A)_{i+j \cdot N} = a_{ij}$ to obtain relabeled angles $(\theta_0, \dots, \theta_3)$. It is clear from the system of equations above that these angles have to satisfy:

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} \hat{\theta}_0 \\ \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \end{bmatrix}, \quad (4.17)$$

the linear system can also be written as

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} \hat{\theta}_0 \\ \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \end{bmatrix}, \quad (4.18)$$

$$= (\hat{H} \otimes \hat{H}) P_G \begin{bmatrix} \hat{\theta}_0 \\ \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \end{bmatrix}, \quad (4.19)$$

where $\hat{H} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ is a scalar multiple of the Hadamard gate and P_G is the permutation matrix that transforms binary ordering to Gray code ordering (an ordering of the binary numeral system such that two consecutive values differ in only one bit). This algorithm generalizes to \mathcal{O}_A oracle for matrices

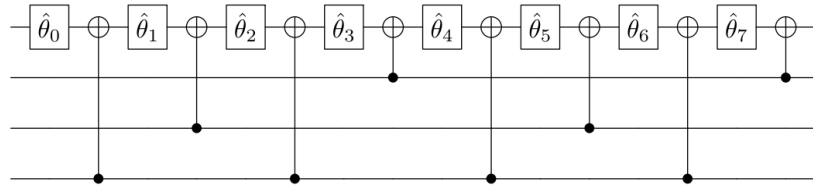
$A \in \mathbb{R}^{N \times N}$. The corresponding circuit structure consists of a gate sequence of length 2^{2n} alternating between R_y and CNOT gates. Note that R_y gates only act on the first qubit, which is also the target of the CNOT gates, and the control qubit for the l -th CNOT is determined by the bit where the l -th and $(l + 1)$ -th Gray code differs. For an oracle \mathcal{O}_A with angles $\boldsymbol{\theta} = (\theta_0, \dots, \theta_{2^{2n}-1})$ given by (4.14), the new angles $\hat{\boldsymbol{\theta}} = (\hat{\theta}_0, \dots, \hat{\theta}_{2^{2n}-1})$ are related to $\boldsymbol{\theta}$ through the linear system

$$(\hat{H}^{\otimes 2n} P_G) \hat{\boldsymbol{\theta}} = \boldsymbol{\theta} \quad (4.20)$$

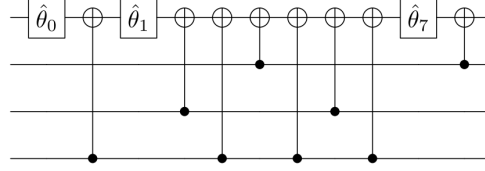
This linear system can be efficiently solved by a classical algorithm in $\mathcal{O}(N^2 \log N^2)$ time using a fast Walsh–Hadamard transform [21]. The gate complexity of implementing \mathcal{O}_A with this approach is $\mathcal{O}(N^2)$ for $A \in \mathbb{R}^{N \times N}$, where N^2 CNOT and N^2 single-qubit R_y gates are required. For the complex case, see [10]. We next illustrate the idea of the circuit compression algorithm for uniformly controlled rotation gates. The compression algorithm uses a cutoff threshold $\delta_c \in \mathbb{R}^+$, that is it considers all $\hat{\theta}_i \leq \delta_c$ to be negligible: this means that the respective single qubit rotations can be removed from the circuit. The algorithm consists of two steps:

1. Remove all rotation gates for angles $\hat{\theta}_i \leq \delta_c$ from the circuit;
2. Perform a parity check on the control qubits of the CNOT gates in each series of consecutive CNOT gates: keep one CNOT gate with control qubit i if there are an odd number in the series, otherwise remove all CNOT gates with control qubit i .

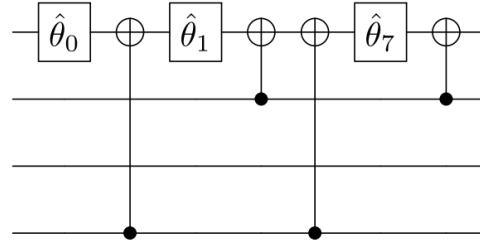
Example 4.2.1. We illustrate the idea of the circuit compression algorithm for controlled rotation gate with 8 angles:



Assume for example that $\hat{\theta}_2, \hat{\theta}_3, \hat{\theta}_4, \hat{\theta}_5, \hat{\theta}_6 \leq \delta_c$, then the respective single qubit rotations can be removed from the circuit, yielding:



The resulting circuit includes a sequence of consecutive CNOT gates that can be simplified, since they commute with each other and act on the same target qubit. Within such a sequence, any pair of CNOT gates sharing the same control qubit will effectively cancel one another out. As a result, in the given example, a total of 5 single-qubit rotations and 4 CNOT gates can be eliminated, leading to a more compact version of the circuit.



This procedure can be considered as data sparsification since it allows us to represent the block encoding matrix with fewer than N^2 parameters. However, since we perform a sparsification on the $\hat{\theta}$ angles after the Wash-Hadamard transform, it doesn't mean that θ and A are sparse in the usual sense of containing many zeros. The following theorem relates the cutoff threshold δ_c with the error ϵ on the block-encoding.

Theorem 4.2.2 ([10]). For an n -qubit matrix $A \in \mathbb{R}^{2^n \times 2^n}$, the FABLE (Fast Approximate Quantum Circuits for Block-Encodings) circuit with cutoff compression threshold $\delta_c \in \mathbb{R}^+$ gives an $(\frac{1}{2^n \cdot \|A\|_\infty}, n + 1, N^3 \delta_c + O(\delta_c^3))$ -block-encoding of A .

Proof. We start with the linear system in eq. (4.19) that relates the angles of the uniformly controlled rotations to the angles of the matrix query oracle. After thresholding the parameters $\hat{\boldsymbol{\theta}}$ with cutoff δ_c , the uniformly controlled rotation is constructed with parameters $\hat{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}$, where $|\delta\theta_i| \leq \delta_c$. It follows that $\|\delta\hat{\boldsymbol{\theta}}\| \leq N\delta_c$. This perturbs the angles in \mathcal{O}_A from $\boldsymbol{\theta}$ to :

$$\tilde{\boldsymbol{\theta}} = (\hat{H}^{\otimes 2n} P_G)(\hat{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}).$$

By linearity, the error on \mathcal{O}_A thus becomes:

$$\boldsymbol{\eta} = \tilde{\boldsymbol{\theta}} - \boldsymbol{\theta} = (\hat{H}^{\otimes 2n} P_G)\delta\boldsymbol{\theta}$$

and we get

$$\|\boldsymbol{\eta}\| \leq \|\hat{H}^{\otimes 2n}\| \cdot \|P_G\| \cdot \|\delta\boldsymbol{\theta}\| \leq N^2\delta_c$$

as P_G is a unitary matrix and $\|\hat{H}^{\otimes 2n}\| = N$. This implies that the element wise error is now only bounded by $|\eta_i| = |\theta_i - \tilde{\theta}_i| \leq N^2\delta_c$. This relates to the element-wise error on a_i as :

$$\begin{aligned} |\delta a_i| &= |a_i - \tilde{a}_i| = |\cos(\theta_i) - \cos(\tilde{\theta}_i)| \\ &= |\cos(\theta_i) - \cos(\theta_i + \eta_i)| \\ &= \left| 2 \sin\left(\frac{\eta_i}{2}\right) \sin\left(\theta_i + \frac{\eta_i}{2}\right) \right| \\ &\leq 2 \left| \sin\left(\frac{\eta_i}{2}\right) \right| \leq N^2\delta_c + \mathcal{O}(\delta_c^3) \end{aligned}$$

In the final approximation, we used the Taylor series of the sine function. We thus have that $\|\delta\mathbf{a}\| \leq N^3\delta_c$. As $\|\mathbf{A}\| \leq \|\mathbf{A}\|_F = \|\text{vec}(\mathbf{A})\|$, we get the upper bound. \square

4.3 Sparse matrices

This section closely follows the constructions presented in [11]. For an alternative construction, we also refer the reader to [22]. We consider the block-encoding of sparse matrices, that is, matrices where most entries are zero. We begin by introducing some notation that will be used throughout.

Definition 4.3.1. A matrix $A \in \mathbb{C}^{n \times m}$ is called *s-sparse* if each row and each column of A contains at most s nonzero entries. In other words,

$$\begin{aligned} \forall i \in \{1, \dots, n\} : & \quad |\{j : A_{ij} \neq 0\}| \leq s, \\ \forall j \in \{1, \dots, m\} : & \quad |\{i : A_{ij} \neq 0\}| \leq s. \end{aligned}$$

Additionally, we introduce a function that returns the location of the ℓ -th nonzero entry in a specified row or column.

Definition 4.3.2. Let $A \in \mathbb{C}^{n \times m}$ be a s -sparse matrix, we define

$$c : \{1, \dots, s\} \times \{1, \dots, m\} \longrightarrow \{1, \dots, n\},$$

as the function that gives the row index of the ℓ -th nonzero matrix element in the j -th column.

We now assume that A is a square matrix, i.e., $A \in \mathbb{C}^{N \times N}$ with $N = 2^n$ and $s = 2^m$ for some $m \ll n$. Under these assumptions, the following theorem provides a method for constructing a block encoding of A .

Theorem 4.3.1 (Theorem 4.1 [11]). Let $c(\ell, j)$ as defined in definition 4.3.2 of an s -sparse matrix $A \in \mathbb{C}^{N \times N}$. Let O_c be a unitary such that:

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

and O_A be a unitary such that

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

then

$$U_A = (I_2 \otimes H^{\otimes m} \otimes I_N)(I_2 \otimes O_c)O_A(I_2 \otimes H^{\otimes m} \otimes I_N),$$

is a $(s, m + 1)$ -block encoding of A .

Remark 4.3.1. The unitaries O_c and O_A in Theorem 4.3.1 always exist. However, what is nontrivial in practice is whether these unitaries can be implemented in an efficient way. The applicability of this block encoding approach ultimately depends on our ability to efficiently construct and execute O_c and O_A for a given matrix A .

Proof. We will show that $\langle 0 | \langle 0^{\otimes m} | \langle i | U_A | 0 \rangle | 0^{\otimes m} \rangle | j \rangle = \frac{1}{s} A_{ij}$. We first apply $H^{\otimes m}, O_A, O_c$ to $|0\rangle |0^{\otimes m}\rangle |j\rangle$:

$$\begin{aligned} |0\rangle |0^{\otimes m}\rangle |j\rangle &\xrightarrow{I_2 \otimes H^{\otimes m} \otimes I_N} \frac{1}{\sqrt{s}} \sum_{\ell=0}^{s-1} |0\rangle |\ell\rangle |j\rangle, \\ &\xrightarrow{O_A} \frac{1}{\sqrt{s}} \sum_{\ell=0}^{s-1} \left(A_{c(\ell,j),j} |0\rangle + \sqrt{1 - |A_{c(\ell,j),j}|^2} |1\rangle \right) |\ell\rangle |j\rangle, \\ &\xrightarrow{I_2 \otimes O_c} \frac{1}{\sqrt{s}} \sum_{\ell=0}^{s-1} \left(A_{c(\ell,j),j} |0\rangle + \sqrt{1 - |A_{c(\ell,j),j}|^2} |1\rangle \right) |\ell\rangle |c(\ell,j)\rangle, \end{aligned}$$

and we apply $I_2 \otimes H^{\otimes m} \otimes I_N$ first to $|0\rangle |0^{\otimes m}\rangle |i\rangle$, that is

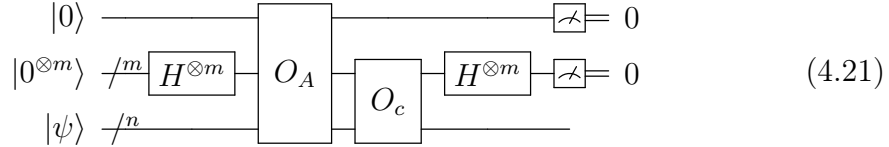
$$|0\rangle |0^{\otimes m}\rangle |i\rangle \xrightarrow{I_2 \otimes H^{\otimes m} \otimes I_N} \frac{1}{\sqrt{s}} \sum_{k=0}^{s-1} |0\rangle |k\rangle |i\rangle.$$

Finally, taking the inner product we get

$$\langle 0 | \langle 0^{\otimes m} | \langle i | U_A | 0 \rangle | 0^{\otimes m} \rangle | j \rangle = \frac{1}{s} \sum_{k=0}^{s-1} \sum_{\ell=0}^{s-1} A_{c(\ell,j),j} \langle k | \ell \rangle \langle i | c(\ell,j) \rangle = \frac{1}{s} A_{ij}$$

□

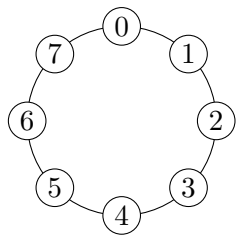
The quantum circuit associated with this block encoding has the following general form:



This block-encoding can be regarded as a special case of Theorem [4.2.1](#). However, due to the sparsity of A , it suffices to use only $m + 1$ ancilla qubits instead of $n + 1$. Moreover, if A possesses additional structure, the oracles O_c and O_A can often be implemented efficiently. In the following, we focus on a specific class of structured matrices and provide explicit constructions of the corresponding quantum circuits.

A sparse matrix A can naturally be interpreted as the adjacency matrix of a directed graph. In this context, we define a vertex set V corresponding to the column indices $\{j\}$ of A , and an edge set $E \subseteq V \times V$ consisting of index

pairs (i, j) such that $A_{ij} \neq 0$. Each nonzero entry A_{ij} represents a directed edge from vertex j to vertex i , with the value of A_{ij} specifying the weight of that edge. In the case of undirected graphs, the corresponding adjacency matrix is symmetric, satisfying $A_{ij} = A_{ji}$. In contrast, for directed graphs, this symmetry condition does not hold in general. Now, we focus our study on the special case where the sparse matrix A corresponds to the adjacency matrix of a directed cyclic graph. As an illustrative example, we consider a cyclic graph with $N = 8$ vertices, whose associated adjacency matrix is a banded circulant matrix.



$$A = \begin{bmatrix} \alpha & \gamma & 0 & \cdots & \beta \\ \beta & \alpha & \ddots & \ddots & 0 \\ 0 & \beta & \ddots & \gamma & \vdots \\ \vdots & \ddots & \ddots & \alpha & \gamma \\ \gamma & 0 & \cdots & \beta & \alpha \end{bmatrix}. \quad (4.22)$$

The structure of the matrix corresponds to a directed graph in which each vertex j , for $0 \leq j \leq N - 1$, has two outgoing edges. Specifically, the edges $(\text{mod}(j + 1, N), j)$ and $(j, \text{mod}(j - 1, N))$ are included in the edge set E . The weights associated with these directed edges are denoted by β and γ , respectively. In addition to these directed edges, each vertex j is assigned a weight α , which can be interpreted as the weight of a self-loop edge from vertex j to itself. As a result, the corresponding adjacency matrix A takes on a nearly tridiagonal structure, with:

- α along the diagonal,
- γ along the superdiagonal,
- β along the subdiagonal.

The cyclic nature of the graph is reflected by the nonzero entries $A_{N-1,0} = \gamma$ and $A_{0,N-1} = \beta$, which complete the wraparound connectivity of the cycle. Each column of the matrix contains exactly three nonzero elements. To

encode the row indices of these nonzero elements, we require $m = \lceil \log_2 3 \rceil = 2$ ancilla qubits. An additional ancilla qubit is used to encode the numerical values of the matrix entries. When applying Theorem 4.3.1 to construct the corresponding block-encoding circuit, we use $s = 2^m = 4$ and thus treat the matrix as 4-sparse, even though each column actually contains only three nonzero elements.

In order to construct an explicit block encoding, we first need to define the function $c(j, \ell)$ introduced in Definition 4.3.2, which returns the row index of the ℓ -th nonzero entry in the j -th column of the matrix. For the adjacency matrix associated with a cyclic graph, this function can be specified explicitly as follows:

$$c(\ell, j) = \begin{cases} (j - 1) \bmod N & \text{if } \ell = 0 \\ j & \text{if } \ell = 1 \text{ or } \ell = 3 \\ (j + 1) \bmod N & \text{if } \ell = 2 \end{cases} \quad (4.23)$$

Therefore we need to construct a quantum circuit to map $|j\rangle$ to $|(j - 1) \bmod N\rangle$, $|j\rangle$ or $|(j + 1) \bmod N\rangle$, depending on the value of ℓ . These cyclic addition and subtraction mappings correspond to left and right shift permutation operators, which we define as follows:

$$L = \begin{bmatrix} 0 & & & 1 \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ 1 & & & 0 \end{bmatrix}. \quad (4.24)$$

Here, the matrix L performs a cyclic left shift (i.e., subtraction by 1 mod N), and R performs a cyclic right shift (i.e., addition by 1 mod N). These operations can be implemented efficiently in quantum circuits using multiqubit controlled-NOT gates.

Lemma 4.3.1. We define the C^n -NOT gate as the n -qubit multi-controlled NOT gate acting as:

$$C^n\text{-NOT } |j_{n-1}\rangle \cdots |j_1\rangle |j_0\rangle = |j_{n-1} \oplus \left(\prod_{k=0}^{n-2} j_k \right)\rangle \cdots |j_1\rangle |j_0\rangle. \quad (4.25)$$

Analogously, we define the negatively-controlled version, denoted \tilde{C}^n -NOT, as:

$$\tilde{C}^n\text{-NOT } |j_{n-1}\rangle \cdots |j_1\rangle |j_0\rangle = |j_{n-1} \oplus \left(\prod_{k=0}^{n-2} (1 \oplus j_k) \right)\rangle \cdots |j_1\rangle |j_0\rangle. \quad (4.26)$$

Then the left and right cyclic shift operators introduced in Eq. (4.24) can be constructed as:

$$L = (I_{2^{n-1}} \otimes X) \cdot \left(\prod_{p=1}^n C^p\text{-NOT} \right), \quad (4.27)$$

$$R = (I_{2^{n-1}} \otimes X) \cdot \left(\prod_{p=1}^n \tilde{C}^p\text{-NOT} \right). \quad (4.28)$$

Then L implements modular addition by 1, and R implements modular subtraction by 1:

$$L|j\rangle = |(j+1) \bmod 2^n\rangle, \quad R|j\rangle = |(j-1) \bmod 2^n\rangle.$$

Proof. We proceed by induction on n . For the base case $n = 1$, we have:

$$L = X,$$

which acts as:

$$L|0\rangle = |1\rangle, \quad L|1\rangle = |0\rangle,$$

corresponding to modular addition modulo 2, as required. Now suppose the claim holds for some $n = k$, i.e., the operator

$$L_k = (I_{2^{k-1}} \otimes X) \left(\prod_{p=1}^k C^p\text{-NOT} \right)$$

implements:

$$L_k|j\rangle = |(j+1) \bmod 2^k\rangle.$$

We now prove the result for $n = k + 1$. Define:

$$L_{k+1} = (I_{2^k} \otimes X) \cdot \left(\prod_{p=1}^{k+1} C^p\text{-NOT} \right) = L_k \cdot C^{k+1}\text{-NOT}.$$

The gate C^{k+1} -NOT is activated only on states of the form $|j_k\rangle |1 \cdots 1\rangle$, where the last k qubits are all 1. These correspond to the numbers of the form:

$$j_k \cdot 2^k + (2^k - 1).$$

On such states, we have:

$$\begin{aligned} L_{k+1} |j_k\rangle |1 \cdots 1\rangle &= L_k \cdot C^{k+1}\text{-NOT} |j_k\rangle |1 \cdots 1\rangle \\ &= L_k |j_k \oplus 1\rangle |1 \cdots 1\rangle \\ &= |j_k \oplus 1\rangle |0 \cdots 0\rangle, \end{aligned}$$

which corresponds to the state $|j_k \cdot 2^k\rangle$, i.e., $(j+1) \bmod 2^{k+1}$. For all other basis states $|j\rangle = |j_k\rangle |j_{k-1}\rangle \cdots |j_0\rangle$, where the last k bits are not all ones, the C^{k+1} -NOT gate is inactive, so:

$$L_{k+1} |j\rangle = L_k |j\rangle = |(j+1) \bmod 2^{k+1}\rangle.$$

Hence, the induction step holds and L_{k+1} performs modular addition by one.

The proof for R is analogous, using negatively-controlled gates instead. \square

In the case $n = 4$, the circuit representation of L is:

$$\begin{array}{c} |j_3\rangle \\ |j_2\rangle \\ |j_1\rangle \\ |j_0\rangle \end{array} \begin{array}{c} \boxed{L} \end{array} = \begin{array}{c} \text{---} \oplus \text{---} \\ \text{---} \bullet \text{---} \oplus \text{---} \\ \text{---} \bullet \text{---} \bullet \text{---} \oplus \text{---} \\ \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \boxed{X} \end{array} \quad (4.29)$$

Similarly, the circuit for R is:

$$\begin{array}{c} |j_3\rangle \\ |j_2\rangle \\ |j_1\rangle \\ |j_0\rangle \end{array} \begin{array}{c} \boxed{R} \end{array} = \begin{array}{c} \text{---} \oplus \text{---} \\ \text{---} \circ \text{---} \oplus \text{---} \\ \text{---} \circ \text{---} \circ \text{---} \oplus \text{---} \\ \text{---} \circ \text{---} \circ \text{---} \circ \text{---} \boxed{X} \end{array} \quad (4.30)$$

To construct O_c we have to control the L and R gates which must be activated only when $\ell = 0$ and $\ell = 2$. The final circuit for O_c is :

$$\begin{array}{c} |\ell_1\rangle \\ |\ell_0\rangle \\ |j\rangle \end{array} \begin{array}{c} \boxed{O_c} \end{array} = \begin{array}{c} \text{---} \circ \text{---} \bullet \text{---} \\ \text{---} \circ \text{---} \circ \text{---} \\ \text{---} \boxed{L} \text{---} \boxed{R} \text{---} \end{array} \quad (4.31)$$

$$c(\ell, j) = \begin{cases} j & \text{if } \ell = 0 \text{ or } \ell = 3 \\ (j-1) \bmod N & \text{if } \ell = 1 \\ (j+1) \bmod N & \text{if } \ell = 2 \end{cases} \quad (4.32)$$
$$\begin{array}{c} |\ell_1\rangle \\ |\ell_0\rangle \\ |j\rangle \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{O_c} \\ \boxed{} \\ \boxed{} \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \circ \\ \bullet \\ \bullet \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \circ \\ \bullet \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{L} \\ \boxed{R} \\ \boxed{} \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{L} \\ \boxed{R} \\ \boxed{} \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \quad (4.33)$$

Lemma 4.3.2. The rotation angles are given by:

$$\theta_0 = 2 \arccos(\alpha - 1), \quad \theta_1 = 2 \arccos(\beta), \quad \theta_2 = 2 \arccos(\gamma).$$

$$\langle 0 | \langle 00 | \langle i | U_A | 0 \rangle | 00 \rangle | j \rangle, \quad \text{for } i \in \{j-1, j, j+1\},$$

where U_A is the operator defined in Theorem 4.3.1. We apply the operator $(I_2 \otimes O_c)O_A(I_2 \otimes H \otimes H \otimes I_n)$ to the state $|0\rangle|00\rangle|j\rangle$, and we apply the operator $I_2 \otimes H \otimes H \otimes I_n$ to the state $|0\rangle|00\rangle|i\rangle$. We then compute their

inner product.

$$\begin{aligned}
& (I_2 \otimes O_c) O_A (I_2 \otimes H \otimes H \otimes I_n) |0\rangle |00\rangle |j\rangle \\
&= \frac{1}{2} (I_2 \otimes O_c) O_A |0\rangle (|00\rangle + |01\rangle + |10\rangle + |11\rangle) |j\rangle \\
&= \frac{1}{2} (I_2 \otimes O_c) (R_y(\theta_0) |0\rangle |00\rangle + R_y(\theta_1) |0\rangle |01\rangle + R_y(\theta_2) |0\rangle |10\rangle + |0\rangle |11\rangle) |j\rangle \\
&= \frac{1}{2} \left(R_y(\theta_0) |0\rangle |00\rangle |j\rangle + R_y(\theta_1) |0\rangle |01\rangle |j-1\rangle + R_y(\theta_2) |0\rangle |10\rangle |j+1\rangle + |0\rangle |11\rangle |j\rangle \right).
\end{aligned} \tag{4.34}$$

On the other hand, we have:

$$|0\rangle |00\rangle |i\rangle \xrightarrow{I_2 \otimes H \otimes H \otimes I_n} \frac{1}{2} |0\rangle (|00\rangle + |01\rangle + |10\rangle + |11\rangle) |i\rangle. \tag{4.35}$$

Now, computing the inner product for $i \in \{j-1, j, j+1\}$ gives:

$$\langle 0 | \langle 00 | \langle i | U_A | 0 \rangle | 00 \rangle | j \rangle = \begin{cases} \cos\left(\frac{\theta_0}{2}\right) + 1 & \text{if } i = j, \\ \cos\left(\frac{\theta_1}{2}\right) & \text{if } i = j-1, \\ \cos\left(\frac{\theta_2}{2}\right) & \text{if } i = j+1. \end{cases}$$

It is then sufficient to solve the following equations:

$$\langle 0 | \langle 00 | \langle i | U_A | 0 \rangle | 00 \rangle | j \rangle = \begin{cases} \alpha & \text{if } i = j, \\ \beta & \text{if } i = j-1, \\ \gamma & \text{if } i = j+1, \end{cases}$$

from which the expressions for θ_0 , θ_1 , and θ_2 follow. \square

This design leads to the following orthogonal array (O_A) quantum circuit structure, where the rotations are activated only for the appropriate values of ℓ to ensure that the correct matrix elements are encoded.

$$\begin{aligned}
& \begin{array}{c} |0\rangle \\ |l_1\rangle \\ |l_0\rangle \\ |j\rangle \end{array} \begin{array}{|c|} \hline O_A \\ \hline \end{array} = \begin{array}{c} \text{---} \boxed{R_y(\theta_0)} \text{---} \boxed{R_y(\theta_1)} \text{---} \boxed{R_y(\theta_2)} \text{---} \\ \text{---} \circ \text{---} \circ \text{---} \bullet \text{---} \\ \text{---} \circ \text{---} \bullet \text{---} \circ \text{---} \\ \text{---} \end{array} \tag{4.36}
\end{aligned}$$

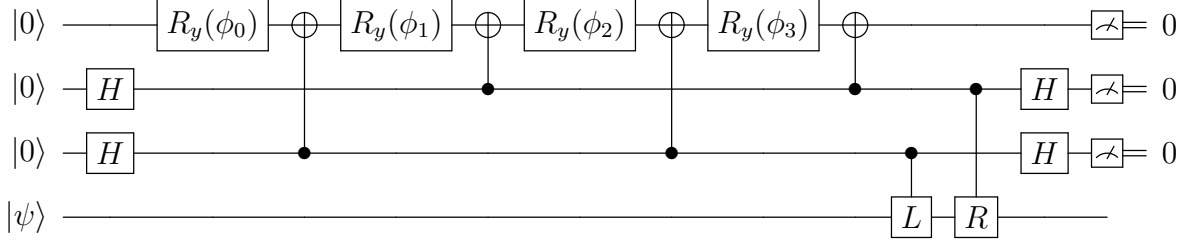
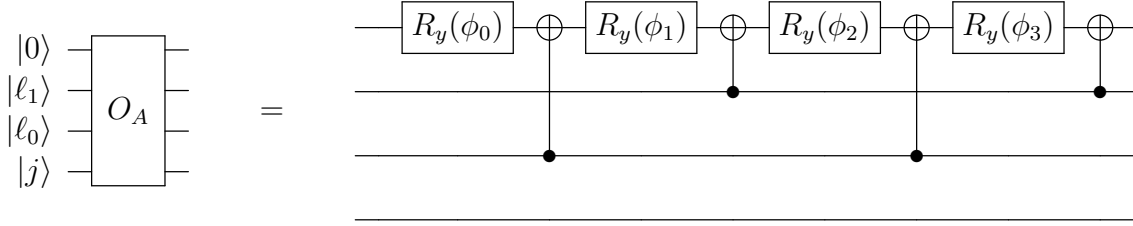


Figure 4.1: A complete quantum circuit for the block encoding of a 8×8 banded circulant matrix.

To avoid the use of multi-controlled gates, we can apply a Walsh–Hadamard transformation of angles, as described in eq. (4.20). Then, the circuit O_A with the angles $\{\phi_0, \phi_1, \phi_2, \phi_3\}$, can be implemented in the following way:



The complete circuit for a block encoding of $A/4$, where A is an 8×8 circulant matrix of the form given in eq. (4.22), is shown in Figure 4.1. The total number of Hadamard gates and controlled rotations scales proportionally with $\log(s)$, which remains small for sparse matrices. Similarly, the number of controlled- R and controlled- L shift operations is $O(\log s)$. Each of these shift operations is implemented as a general multi-qubit controlled (Toffoli) gate, which can be decomposed into a polynomial number of two-qubit gates. Consequently, the overall gate complexity of the U_A circuit is $\text{poly}(n)$, making the construction asymptotically efficient.

Chapter 5

Efficient Block Encoding of Spin Hamiltonians

In this chapter, we present the main result of the thesis: an efficient framework for the block encoding of spin Hamiltonians, a class of matrices characterized by their structured sparsity and symmetry. We introduce a novel formulation of the Linear Combination of Unitaries (LCU) method that the SELECT oracle using only n CNOT and n cZ gates, along with a linear number of ancilla qubits. By carefully exploiting the structure inherent to spin Hamiltonians, we also reduce the cost of the state preparation oracle. This is achieved through the design of a new family of efficient routines for Dicke state preparation, a special class of quantum states.

To demonstrate the practicality of our framework, we explicitly construct block-encoding circuits for a representative case: the Heisenberg Hamiltonian. We provide detailed, non-asymptotic gate counts for all components of the circuit and support our analysis with numerical benchmarks. These results validate the effectiveness of our method—referred to as the FOQCS-LCU technique—by showing a substantial reduction in CNOT gate count, achieving an order-of-magnitude improvement over conventional LCU-based approaches.

5.1 Fast One-Qubit Control Select LCU

In the Linear Combination of Unitaries (LCU) method (see Remark 4.1.2), a target matrix H is expressed as a weighted sum of unitary operators:

$$H = \sum_{m=0}^{M-1} \alpha_m U_m, \quad (5.1)$$

where the α_m are (generally complex) scalar coefficients, and each U_m is a unitary matrix.

This decomposition provides a flexible and general framework for encoding arbitrary operators. While in the worst case the number of terms M may grow exponentially with system size, in many physically relevant settings (for example condensed matter systems and molecular Hamiltonians in quantum chemistry) M scales only polynomially, or even linearly. Due to its generality and scalability in structured problems, the LCU method remains one of the most widely used and theoretically grounded strategies for implementing efficient block encodings in quantum simulation tasks.

We now proceed to examine the standard LCU method in detail, before introducing our variant. We will highlight the key differences between our approach and the original formulation presented above. The LCU method utilizes three oracles: a state preparation pair (P_R, P_L) (Def 4.1.2) and SELECT. P_R and P_L only act on the $\lceil \log_2(M) \rceil$ ancillae and prepare respectively the following states:

$$|0^{\otimes \lceil \log_2(M) \rceil}\rangle \xrightarrow{P_R} \frac{1}{\sqrt{N}} \sum_{m=0}^{M-1} \sqrt{\alpha_m} |m\rangle, \quad (5.2)$$

$$|0^{\otimes \lceil \log_2(M) \rceil}\rangle \xrightarrow{P_L} \frac{1}{\sqrt{N}} \sum_{m=0}^{M-1} \sqrt{\alpha_m^*} |m\rangle, \quad (5.3)$$

where $|m\rangle$ denotes the computational basis state corresponding to the binary representation of the integer m and N is the normalization:

$$N = \|\alpha\|_1 = \sum_{m=0}^{M-1} |\alpha_m|. \quad (5.4)$$

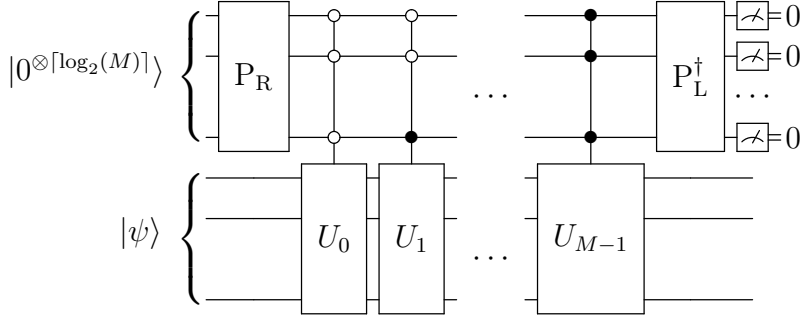


Figure 5.1: LCU block encoding

The SELECT oracle, on the other hand, applies M multi-controlled unitaries acting on the system registry, with each unitary controlled by the associated ancillary state $|m\rangle$:

$$\text{SELECT} = \sum_{m=0}^{M-1} |m\rangle \langle m| \otimes U_m + \sum_{m=M}^{\lceil \log_2(M) \rceil - 1} |m\rangle \langle m| \otimes I \quad (5.5)$$

The complete LCU circuit starts with the preparation of the state eq. (5.2), followed by the application of SELECT, and then the inverse of P_L , as shown in Fig 5.1.

Note that the main difference between this method and the block encoding presented in eq. (4.9) lies in the absence of ancillary qubits $|0^{\otimes a}\rangle$ in the current formulation. These ancillae are not required here, since the operators U_m are already unitary.

If we consider the most generic implementation of LCU, assuming for simplicity that U consists of Pauli strings (i.e tensor product of Pauli matrices), the overall gate count scales as $\mathcal{O}(\omega M \log^\beta(M))$ [9, 23], where ω is the maximum Pauli weight (i.e the maximum number of Pauli matrices that are not the identity in the tensor product) and the greatest contribution is given by the multi-controlled gates of the SELECT oracle. However, this asymptotic cost can be substantially reduced by introducing additional ancillae and exploiting the algebraic or physical structure of the problem at hand. We contribute to this line of research by fully exploiting the underlying structures of target Hamiltonians, particularly those within a broad

Pauli	(i, j)
I	(0, 0)
Z	(0, 1)
X	(1, 0)
Y	(1, 1)

Table 5.1: check-matrix formalism.

class of quantum spin models. Moreover, we develop a novel formulation of LCU that eliminates the need for costly multi-controlled operations in the SELECT oracle.

Finally, we introduce a new LCU implementation circuit based on the check-matrix formalism [1], where we assume the unitaries to be Pauli strings. Since the Pauli matrices form a basis for the space of 2×2 complex matrices, such a decomposition can always be found. The key feature of this new LCU variant is that the SELECT subroutine requires no multi-controlled gates. Therefore, we refer to this method as Fast One-Qubit Control Select LCU (FOQCS-LCU), pronounced “Focus”.

First we present a simple example of a generic 2×2 matrix, to illustrate how our implementation works. After that, we describe how to extend this approach to general matrices of dimension $2^n \times 2^n$.

We start with the single-qubit case of any complex matrix $A \in \mathbb{C}^{2 \times 2}$ which can be written as:

$$A = \alpha_{00}I + \alpha_{01}Z + \alpha_{10}X + \alpha_{11}Y, \quad (5.6)$$

$$= \alpha_{00}I + \alpha_{01}Z + \alpha_{10}X - i\alpha_{11}ZX, \quad (5.7)$$

where we assume $\|\alpha\|_1 = 1$ and use the identity $Y = -iZX$, noting that the coefficients α are generally complex. This notation recalls the check-matrix formalism in table 5.1 [1], so that the coefficient α_{ij} corresponds to the pair (i, j) .

By using a state preparation pair (P_R, P_L) , such that each prepares the

following state on two ancillae:

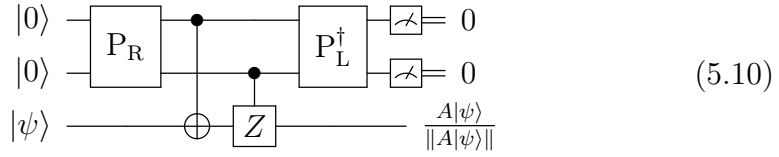
$$|00\rangle \xrightarrow{P_R} \sqrt{\alpha_{00}} |00\rangle + \sqrt{\alpha_{01}} |01\rangle + \sqrt{\alpha_{10}} |10\rangle + \sqrt{-i\alpha_{11}} |11\rangle , \quad (5.8)$$

$$|00\rangle \xrightarrow{P_L} \sqrt{\alpha_{00}^*} |00\rangle + \sqrt{\alpha_{01}^*} |01\rangle + \sqrt{\alpha_{10}^*} |10\rangle + \sqrt{-i\alpha_{11}^*} |11\rangle , \quad (5.9)$$

where the square root of a complex number $z \in \mathbb{C}$ as its principal value, denoted by \sqrt{z} , and given by

$$\sqrt{z} = \sqrt{|z|} e^{i \arg(z)/2} ,$$

where $|z|$ is the modulus of z and $\arg(z) \in (-\pi, \pi]$ is its principal argument. Then, we can realize a modified LCU block encoding for A , where the X and Z contributions are activated respectively by the first and second ancilla:



Theorem 5.1.1. The circuit in eq. (5.10) is a $(2, 0)$ block-encoding for A in eq. (5.6).

Proof. Given a starting one-qubit state $|\psi\rangle$ on which A is applied:

$$A |\psi\rangle = (\alpha_{00}I + \alpha_{01}Z + \alpha_{10}X - i\alpha_{11}ZX) |\psi\rangle ,$$

the circuit yields the following outcomes at each step:

$$\begin{aligned} |00\rangle |\psi\rangle &\xrightarrow{P_R} (\sqrt{\alpha_{00}} |00\rangle + \sqrt{\alpha_{01}} |01\rangle + \sqrt{\alpha_{10}} |10\rangle + \sqrt{-i\alpha_{11}} |11\rangle) |\psi\rangle \\ &\xrightarrow{\text{CNOT}} \sqrt{\alpha_{00}} |00\rangle |\psi\rangle + \sqrt{\alpha_{01}} |01\rangle X |\psi\rangle + \sqrt{\alpha_{10}} |10\rangle |\psi\rangle + \sqrt{-i\alpha_{11}} X |11\rangle |\psi\rangle \\ &\xrightarrow{\text{cZ}} \sqrt{\alpha_{00}} |00\rangle |\psi\rangle + \sqrt{\alpha_{01}} |01\rangle X |\psi\rangle + \sqrt{\alpha_{10}} |10\rangle Z |\psi\rangle + \sqrt{-i\alpha_{11}} ZX |11\rangle |\psi\rangle . \end{aligned}$$

Finally, before measuring the ancillae, we apply the gate P_L^\dagger , whose action is defined as:

$$|00\rangle \xrightarrow{P_L} \sqrt{\alpha_{00}^*} |00\rangle + \sqrt{\alpha_{01}^*} |01\rangle + \sqrt{\alpha_{10}^*} |10\rangle + \sqrt{-i\alpha_{11}^*} |11\rangle ,$$

That is, by applying the conjugate transpose, we obtain:

$$\langle 00 | P_L^\dagger = \sqrt{\alpha_{00}} \langle 00 | + \sqrt{\alpha_{01}} \langle 01 | + \sqrt{\alpha_{10}} \langle 10 | + \sqrt{-i\alpha_{11}} \langle 11 | ,$$

since the complex conjugate of the complex conjugate of α is α itself, so we get:

$$\begin{aligned} (\langle 00 | \otimes I_2) \cdot P_L^\dagger \cdot \text{SELECT} \cdot P_R |00\rangle |\psi\rangle &= \alpha_{00} |\psi\rangle + \alpha_{01} X |\psi\rangle + \alpha_{10} Z |\psi\rangle - i\alpha_{11} ZX |\psi\rangle \\ &= A |\psi\rangle . \end{aligned}$$

□

The circuit in eq. (5.10) can be easily generalized for n qubits for any linear combination of Pauli strings. Given the standard LCU decomposition in Pauli strings:

$$H = \sum_{m=0}^{M-1} \alpha_m \bigotimes_{\ell=0}^{n-1} \sigma_{m_\ell} \quad (5.11)$$

where $\sigma_{m_\ell} \in \{I, X, Y, Z\}$ and $\|\alpha\|_1 = 1$, we can map every σ_{m_ℓ} to a tuple $(\mathbf{i}_\ell, \mathbf{j}_\ell)$ by following the check-matrix formalism in table 5.1. As a result, we get the following identity:

$$\sigma_{m_\ell} = (-i)^{\mathbf{i}_\ell \cdot \mathbf{j}_\ell} Z^{\mathbf{j}_\ell} X^{\mathbf{i}_\ell} , \quad (5.12)$$

where $X^{\mathbf{i}_\ell}$ and $Z^{\mathbf{j}_\ell}$ are defined as:

$$X^{\mathbf{i}_\ell} = \begin{cases} I & \text{if } \mathbf{i}_\ell = 0 \\ X & \text{if } \mathbf{i}_\ell = 1 \end{cases} \quad (5.13)$$

Next, we combine the indices \mathbf{i}_ℓ and \mathbf{j}_ℓ for $\ell = 0, \dots, n-1$ to get the binary representation of two indices i and j as in eq. (2.1).

$$i = [\mathbf{i}_0, \mathbf{i}_1, \dots, \mathbf{i}_{n-1}] , \quad j = [\mathbf{j}_0, \mathbf{j}_1, \dots, \mathbf{j}_{n-1}] , \quad (5.14)$$

so that every m in eq. (5.11) corresponds to a tuple of integers (i, j) .

Finally, by expanding the summation over every i and $j \in \{0, \dots, 2^n - 1\}$ (where $\alpha_{ij} = 0$ if there is no corresponding α_m in the original summation) and by absorbing the complex coefficients:

$$\tilde{\alpha}_{ij} = (-i)^{\sum_{\ell} \mathbf{i}_{\ell} \cdot \mathbf{j}_{\ell}} \alpha_{ij} \quad (5.15)$$

we get the check-matrix LCU decomposition of H :

$$H = \sum_{i=0}^{2^{n-1}} \sum_{j=0}^{2^{n-1}} \tilde{\alpha}_{ij} \bigotimes_{\ell=0}^{n-1} Z^{\mathbf{j}_{\ell}} X^{\mathbf{i}_{\ell}} . \quad (5.16)$$

Note that even if we are now summing over $4^n \geq M$ terms, the normalization constraint stays the same as the new coefficients $\tilde{\alpha}_{ij}$ are either equal to the old ones up to a complex phase or to 0.

Starting from eq. (5.16), we can then define the action of P_R for the n -qubit case:

$$P_R |0^{\otimes n}\rangle |0^{\otimes n}\rangle = \sum_{i=0}^{2^{n-1}} \sum_{j=0}^{2^{n-1}} \sqrt{\tilde{\alpha}_{ij}} |i\rangle |j\rangle , \quad (5.17)$$

while P_L prepares the complex conjugate:

$$P_L |0^{\otimes n}\rangle |0^{\otimes n}\rangle = \sum_{i=0}^{2^{n-1}} \sum_{j=0}^{2^{n-1}} \sqrt{\tilde{\alpha}_{ij}^*} |i\rangle |j\rangle . \quad (5.18)$$

Subsequently, the SELECT oracle can simply be realized by n CNOT and n cZ gates, as shown in Fig 5.2.

Theorem 5.1.2. The circuit in Fig 5.2 with P_R defined in eq. (5.17) and P_L defined in eq. (5.18) is a $(2n, 0)$ -block encoding for H defined in eq. (5.16).

Proof. First we fix the two n -qubit ancillary registers as $|i\rangle = |\mathbf{i}_0, \dots, \mathbf{i}_{n-1}\rangle$ and $|j\rangle = |\mathbf{j}_0, \dots, \mathbf{j}_{n-1}\rangle$. Applying the n CNOTs of the SELECT oracle will result in the following state:

$$|\mathbf{i}_0, \dots, \mathbf{i}_{n-1}\rangle |\mathbf{j}_0, \dots, \mathbf{j}_{n-1}\rangle (X^{\mathbf{i}_0} \otimes X^{\mathbf{i}_1} \otimes \dots \otimes X^{\mathbf{i}_{n-1}}) |\psi\rangle . \quad (5.19)$$

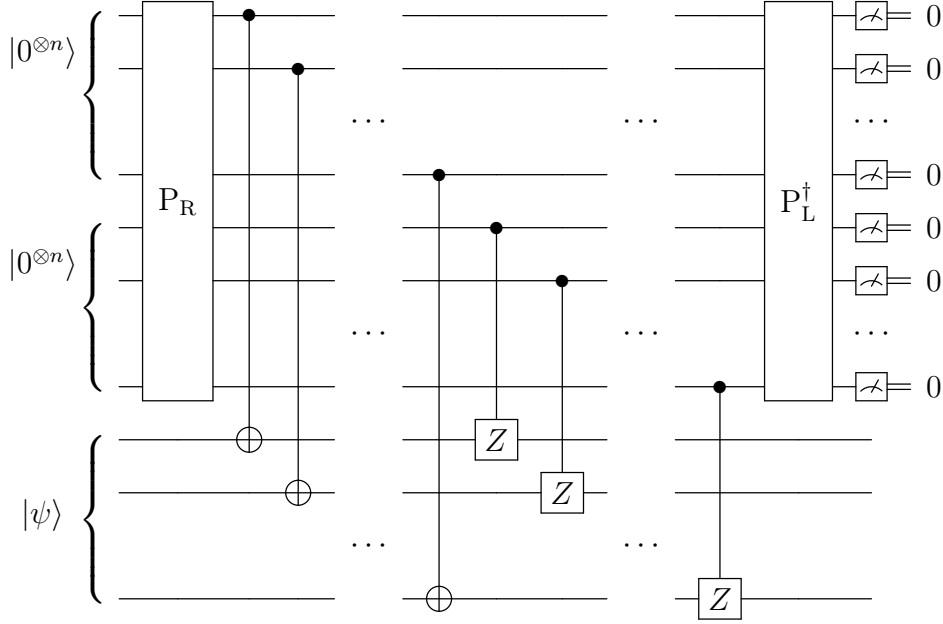


Figure 5.2: Circuits representing FOQCS-LCU.

Then, after the controlled Z , we get:

$$\begin{aligned}
 &\longrightarrow |i_0, \dots, i_{n-1}\rangle |j_0, \dots, j_{n-1}\rangle (Z^{j_0} X^{i_0} \otimes Z^{j_1} X^{i_1} \otimes \dots \otimes Z^{j_{n-1}} X^{i_{n-1}}) |\psi\rangle \\
 &= |i\rangle |j\rangle \bigotimes_{\ell=0}^{n-1} Z^{j_\ell} X^{i_\ell} |\psi\rangle, \tag{5.20}
 \end{aligned}$$

which proves the result for fixed $|i\rangle$ and $|j\rangle$.

If we now consider the P_R and P_L^\dagger gates, whose actions were defined respectively in eq. (5.2) and eq. (5.3), we can now prove that the circuit in

Fig 5.2 implements the block encoding of A :

$$\begin{aligned}
& \langle 0^{\otimes n} | \langle 0^{\otimes n} | P_L^\dagger \cdot \text{SELECT} \cdot P_R | 0^{\otimes n} \rangle | 0^{\otimes n} \rangle |\psi\rangle \\
&= \left[\sum_{i'=0}^{2^{n-1}} \sum_{j'=0}^{2^{n-1}} \sqrt{\tilde{\alpha}_{i'j'}} \langle i' | \langle j' | \right] \text{SELECT} \left[\sum_{i=0}^{2^{n-1}} \sum_{j=0}^{2^{n-1}} \sqrt{\tilde{\alpha}_{ij}} |i\rangle |j\rangle \right] |\psi\rangle \\
&= \left[\sum_{i'=0}^{2^{n-1}} \sum_{j'=0}^{2^{n-1}} \sqrt{\tilde{\alpha}_{i'j'}} \langle i' | \langle j' | \right] \left[\sum_{i=0}^{2^{n-1}} \sum_{j=0}^{2^{n-1}} \sqrt{\tilde{\alpha}_{ij}} |i\rangle |j\rangle \bigotimes_{\ell=0}^{n-1} Z^{j_\ell} X^{i_\ell} \right] |\psi\rangle \\
&= \sum_{i=0}^{2^{n-1}} \sum_{j=0}^{2^{n-1}} \tilde{\alpha}_{ij} \bigotimes_{\ell=0}^{n-1} Z^{j_\ell} X^{i_\ell} |\psi\rangle \\
&= A |\psi\rangle
\end{aligned} \tag{5.21}$$

□

A key advantage of this implementation is that it requires no multi-controlled gates, achieves constant circuit depth (equal to 2), and is completely independent of the specific matrix being encoded. This marks a significant departure from standard LCU constructions, where SELECT is often the most resource-intensive component due to the overhead of decomposing multi-controlled unitaries into elementary gates. In contrast, our approach shifts all matrix-specific complexity to the state preparation oracles, P_L and P_R . These oracles consistently require $2n$ ancillae, compared to the $\lceil \log_2(M) \rceil$ ancillae needed in standard LCU. In the worst-case scenario, where $M = 4^n$, the ancilla requirements and computational costs of both methods become comparable. However, in many physically relevant cases where M is small, our method results in a significantly sparser representation of the new coefficients $\tilde{\alpha}$. A natural first step toward simplifying the implementation of P_R and P_L is to leverage quantum state preparation algorithms tailored for sparse input states. These methods offer complexities ranging from $\mathcal{O}(Mn)$ to $\mathcal{O}(Mn/\log n)$, provided that additional ancillary qubits are available [24, 25, 26]. Moreover, if the target Hamiltonian exhibits additional structure, this can be systematically exploited to further reduce

the cost of state preparation. We demonstrate this in the following sections for the Heisenberg and spin glass Hamiltonians. In summary, our method replaces the costly SELECT oracle of standard LCU with a highly efficient, structure-independent implementation of constant depth.

5.2 Preparation of Dicke states

Before exploring the practical applications of the FOQCS-LCU block encoding, we introduce a set of foundational subroutines that will be used throughout the remainder of this chapter. In particular, we highlight key results from the literature on the preparation of Dicke states $|D_k^n\rangle$ —a family of quantum states that are symmetric under qubit permutations and have exactly ν excitations (i.e qubit in the $|1\rangle$ state) among n qubits [12, 27].

Both the standard Dicke states and certain newly derived variants will play a crucial role in the construction of the P_R and P_L oracles for the spin Hamiltonians.

Definition 5.2.1. The Dicke state $|D_1^n\rangle$, also known as the W -state, is defined as:

$$\begin{aligned} |D_1^n\rangle &= \frac{1}{\sqrt{n}} (|100\dots\rangle + |010\dots\rangle + \dots + |\dots 001\rangle), \\ &= \frac{1}{\sqrt{n}} \sum_{\ell=0}^{n-1} |2^\ell\rangle. \end{aligned} \quad (5.22)$$

To construct the corresponding circuit, we rely on two auxiliary components: the $\Gamma(\theta)$ and Δ^n subroutines.

Definition 5.2.2. The $\Gamma(\theta)$ subroutine is defined as :

$$\text{Circuit diagram for } \Gamma(\theta) = \text{Circuit diagram for } R_y(\theta) \text{ followed by a CNOT gate.} \quad (5.23)$$

and it acts on the basis state in the following way:

$$\begin{aligned} |00\rangle &\xrightarrow{\Gamma(\theta)} |00\rangle & |01\rangle &\xrightarrow{\Gamma(\theta)} \cos \frac{\theta}{2} |01\rangle + \sin \frac{\theta}{2} |10\rangle \\ |10\rangle &\xrightarrow{\Gamma(\theta)} |11\rangle & |11\rangle &\xrightarrow{\Gamma(\theta)} -\sin \frac{\theta}{2} |01\rangle + \cos \frac{\theta}{2} |10\rangle \end{aligned} \quad (5.24)$$

Lemma 5.2.1. The $\Gamma(\theta)$ subroutine can be implemented by using just 2 CNOT with the following circuit:

$$(5.25)$$

Proof. We now show that the two circuits implement the same transformation on computational basis states. Now we compute the action of the circuit in eq. (5.25) on the basis states. Let start with $|00\rangle$, we have:

$$\begin{aligned} |00\rangle &\xrightarrow{(R_z(\frac{\pi}{2}-\frac{\theta}{2}) \cdot H \cdot S) \otimes I_2} \frac{e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |00\rangle + \frac{e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |10\rangle \\ &\xrightarrow{\text{CNOT}} \frac{e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |00\rangle + \frac{e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |11\rangle \\ &\xrightarrow{I_2 \otimes R_z(\frac{\theta}{2}-\frac{\pi}{2})} e^{-i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |00\rangle + e^{i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |11\rangle, \end{aligned}$$

we observe that $e^{-i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |00\rangle + e^{i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |11\rangle = \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$, so

$$\begin{aligned} \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle &\xrightarrow{H \otimes H} \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \\ &\xrightarrow{I_2 \otimes S^*} \frac{1}{\sqrt{2}} |00\rangle - \frac{i}{\sqrt{2}} |11\rangle \\ &\xrightarrow{\text{CNOT}} \frac{1}{\sqrt{2}} |00\rangle - \frac{i}{\sqrt{2}} |10\rangle \\ &\xrightarrow{S \otimes I_2} \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |10\rangle \\ &\xrightarrow{H \otimes I_2} |00\rangle. \end{aligned} \quad (5.26)$$

For $|10\rangle$ we have, in a similar way:

$$\begin{aligned} |10\rangle &\xrightarrow{(R_z(\frac{\pi}{2}-\frac{\theta}{2}) \cdot H \cdot S) \otimes I_2} i \frac{e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |00\rangle - i \frac{e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |10\rangle \\ &\xrightarrow{\text{CNOT}} i \frac{e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |00\rangle - i \frac{e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |11\rangle \\ &\xrightarrow{I_2 \otimes R_z(\frac{\theta}{2}-\frac{\pi}{2})} e^{-i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{i \cdot e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |00\rangle - e^{i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{i \cdot e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |11\rangle, \end{aligned}$$

where $e^{-i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{i \cdot e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |00\rangle - e^{i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{i \cdot e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |11\rangle = \frac{i}{\sqrt{2}} |00\rangle - \frac{i}{\sqrt{2}} |11\rangle$, so

$$\begin{aligned}
\frac{i}{\sqrt{2}} |00\rangle - \frac{i}{\sqrt{2}} |11\rangle &\xrightarrow{H \otimes H} \frac{i}{\sqrt{2}} |01\rangle + \frac{i}{\sqrt{2}} |10\rangle \\
&\xrightarrow{I_2 \otimes S^*} \frac{1}{\sqrt{2}} |01\rangle - \frac{i}{\sqrt{2}} |10\rangle \\
&\xrightarrow{\text{CNOT}} \frac{1}{\sqrt{2}} |01\rangle - \frac{i}{\sqrt{2}} |11\rangle \\
&\xrightarrow{S \otimes I_2} \frac{1}{\sqrt{2}} |01\rangle - \frac{1}{\sqrt{2}} |11\rangle \\
&\xrightarrow{H \otimes I_2} |11\rangle .
\end{aligned} \tag{5.27}$$

For $|01\rangle$ we get:

$$\begin{aligned}
|01\rangle &\xrightarrow{(R_z(\frac{\pi}{2}-\frac{\theta}{2}) \cdot H \cdot S) \otimes I_2} \frac{e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |01\rangle + \frac{e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |11\rangle \\
&\xrightarrow{\text{CNOT}} \frac{e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |01\rangle + \frac{e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |10\rangle \\
&\xrightarrow{I_2 \otimes R_z(\frac{\theta}{2}-\frac{\pi}{2})} e^{i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |01\rangle + e^{-i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |10\rangle ,
\end{aligned}$$

we have that $e^{i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{e^{-i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |01\rangle + e^{-i(\frac{\theta}{4}-\frac{\pi}{4})} \cdot \frac{e^{i(\frac{\pi}{4}-\frac{\theta}{4})}}{\sqrt{2}} |10\rangle = \frac{-i \cdot e^{i\frac{\theta}{2}}}{\sqrt{2}} |01\rangle + \frac{i \cdot e^{-i\frac{\theta}{2}}}{\sqrt{2}} |10\rangle$, so

$$\begin{aligned}
\frac{-i \cdot e^{i\frac{\theta}{2}}}{\sqrt{2}} |01\rangle + \frac{i \cdot e^{-i\frac{\theta}{2}}}{\sqrt{2}} |10\rangle &\xrightarrow{H \otimes H} \frac{1}{\sqrt{2}} \left(\sin \frac{\theta}{2} |00\rangle + i \cos \frac{\theta}{2} |01\rangle - i \cos \frac{\theta}{2} |10\rangle - \sin \frac{\theta}{2} |11\rangle \right) \\
&\xrightarrow{I_2 \otimes S^*} \frac{1}{\sqrt{2}} \left(\sin \frac{\theta}{2} |00\rangle + \cos \frac{\theta}{2} |01\rangle - i \cos \frac{\theta}{2} |10\rangle + i \sin \frac{\theta}{2} |11\rangle \right) \\
&\xrightarrow{\text{CNOT}} \frac{1}{\sqrt{2}} \left(\sin \frac{\theta}{2} |00\rangle + \cos \frac{\theta}{2} |01\rangle - i \cos \frac{\theta}{2} |11\rangle + i \sin \frac{\theta}{2} |10\rangle \right) \\
&\xrightarrow{S \otimes I_2} \frac{1}{\sqrt{2}} \left(\sin \frac{\theta}{2} |00\rangle + \cos \frac{\theta}{2} |01\rangle + \cos \frac{\theta}{2} |11\rangle - \sin \frac{\theta}{2} |10\rangle \right) \\
&\xrightarrow{H \otimes I_2} \cos \frac{\theta}{2} |01\rangle + \sin \frac{\theta}{2} |10\rangle .
\end{aligned} \tag{5.28}$$

Remark 5.2.1. We remark that Δ^n exhibits a recursive structure:

$$\Delta^n = \dots \Gamma(\theta_n) \Delta^{n-1} \quad (5.32)$$

Theorem 5.2.1 ([27]). The Dicke state $|D_1^n\rangle$ can then be prepared by leveraging these subroutines, using the following circuit :

$$|0^{\otimes n}\rangle \xrightarrow{D_1^n} |D_1^n\rangle \quad (5.33)$$

where:

$$\text{---}^n \boxed{D_1^n} \text{---} = \text{---}^{n-1} \boxed{X} \boxed{\Delta^n} \text{---} \quad (5.34)$$

Proof. We prove the statement by induction. For $n = 1$, after applying the X gate to $|0\rangle$, we obtain the state $|1\rangle$, which is precisely $|D_1^1\rangle$. Assume the statement holds for $n = k$. We now prove it for $n = k + 1$. Starting with the state $|0^{\otimes k+1}\rangle$, applying the X gate yields $|0^{\otimes k}\rangle |1\rangle$. Next, after applying $\Gamma(\theta_{k+1})$, the state becomes

$$\cos \frac{\theta_{k+1}}{2} |0^{\otimes k-1}\rangle |01\rangle + \sin \frac{\theta_{k+1}}{2} |0^{\otimes k-1}\rangle |10\rangle .$$

By definition, $\theta_{k+1} = 2 \arccos \sqrt{\frac{1}{k+1}}$, so that

$$\cos \frac{\theta_{k+1}}{2} = \sqrt{\frac{1}{k+1}} \quad \text{and} \quad \sin \frac{\theta_{k+1}}{2} = \sqrt{\frac{k}{k+1}} .$$

Thus, the state becomes

$$\frac{1}{\sqrt{k+1}} |0^{\otimes k}\rangle |1\rangle + \sqrt{\frac{k}{k+1}} |0^{\otimes k-1}\rangle |1\rangle |0\rangle .$$

Now, applying the induction hypothesis to the state $|0^{\otimes k-1}\rangle |1\rangle$, we obtain

$$|D_1^k\rangle = \frac{1}{\sqrt{k}} \left(|100 \dots\rangle + |010 \dots\rangle + \dots + |\dots 001\rangle \right) .$$

Therefore, the final state is

$$\frac{1}{\sqrt{k+1}} |0^{\otimes k}\rangle |1\rangle + \sqrt{\frac{k}{k+1}} \frac{1}{\sqrt{k}} \left(|100\cdots\rangle + |010\cdots\rangle + \cdots + |\cdots 001\rangle \right) |0\rangle ,$$

which is exactly $|D_1^{k+1}\rangle$. \square

Moreover, this circuit can be generalized to represent any unbalanced superposition of states with exactly $\nu = 1$ excitations. We denote as *unbalanced* Dicke states:

$$|D_1^n(\alpha)\rangle = \sum_{\ell=0}^{n-1} \alpha_\ell |2^\ell\rangle \quad (5.35)$$

with $\|\alpha\| = 1$. We start by considering only the absolute value of $\alpha_\ell = e^{i\eta_\ell} |\alpha_\ell|$, where the phase factor is fixed at a later stage.

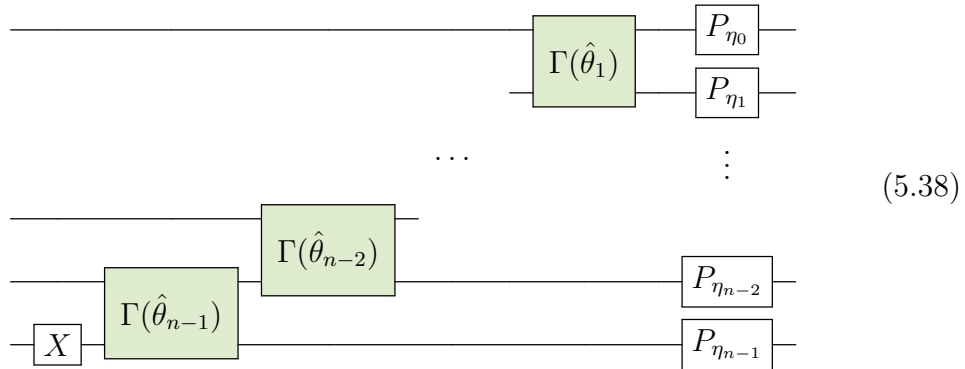
In particular, we prove that by defining the angles θ_ℓ within the circuit in eq. (5.33) as:

$$\hat{\theta}_{n-1} = 2 \arccos |\alpha_{n-1}| \quad (5.36)$$

$$\hat{\theta}_\ell = 2 \arccos \frac{|\alpha_\ell|}{\sqrt{1 - \sum_{j=\ell+1}^{n-1} |\alpha_j|^2}} \quad \ell = n-2, \dots, 1 \quad (5.37)$$

we obtain the state in eq. (5.35) up to some relative phases.

Corollary 5.2.1. The *unbalanced* Dicke state defined in eq. (5.35) can be implemented by the following circuit:



Proof. First of all, it is important to observe the outcome of the CNOT and controlled R_y couple on the initial two-qubit state $|01\rangle$:

$$\begin{aligned} |\cdots 01\rangle &\longrightarrow \cos \frac{\hat{\theta}_{n-1}}{2} |\cdots 01\rangle + \sin \frac{\hat{\theta}_{n-1}}{2} |\cdots 10\rangle \\ &= |\alpha_{n-1}| |2^{n-1}\rangle + \sqrt{1 - |\alpha_{n-1}|^2} |2^{n-2}\rangle . \end{aligned} \quad (5.39)$$

Subsequently, we get to the second step:

$$\begin{aligned} &\longrightarrow |\alpha_{n-1}| |\cdots 001\rangle + \cos \frac{\hat{\theta}_{n-2}}{2} \sqrt{1 - |\alpha_{n-1}|^2} |\cdots 010\rangle + \sin \frac{\hat{\theta}_{n-2}}{2} \sqrt{1 - |\alpha_{n-1}|^2} |\cdots 100\rangle \\ &= |\alpha_{n-1}| |2^{n-1}\rangle + |\alpha_{n-2}| |2^{n-2}\rangle + \sqrt{1 - |\alpha_{n-2}|^2 - |\alpha_{n-1}|^2} |2^{n-3}\rangle , \end{aligned} \quad (5.40)$$

where we have used the following relations:

$$\cos \frac{\hat{\theta}_{n-2}}{2} = \frac{|\alpha_{n-2}|}{\sqrt{1 - |\alpha_{n-1}|^2}} , \quad (5.41)$$

$$\sin \frac{\hat{\theta}_{n-2}}{2} = \sqrt{1 - \left(\cos \frac{\hat{\theta}_{n-2}}{2} \right)^2} = \sqrt{1 - \frac{|\alpha_{n-2}|^2}{1 - |\alpha_{n-1}|^2}} = \frac{\sqrt{1 - |\alpha_{n-2}|^2 - |\alpha_{n-1}|^2}}{\sqrt{1 - |\alpha_{n-1}|^2}} . \quad (5.42)$$

In a recursive manner, we can then extract the final state:

$$\longrightarrow \sqrt{1 - \sum_{\ell=1}^{n-1} |\alpha_\ell|^2} |2^0\rangle + \sum_{\ell=0}^{n-2} |\alpha_\ell| |2^\ell\rangle , \quad (5.43)$$

where it is sufficient to observe that:

$$\sqrt{1 - \sum_{\ell=1}^{n-1} |\alpha_\ell|^2} = |\alpha_0| , \quad (5.44)$$

since $\|\alpha\|_2 = 1$, to get the state in eq. (5.35) up to some relative phases.

Finally, in order to correct the phases, it is sufficient to apply a phase gate:

$$P_{\eta_\ell} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\eta_\ell} \end{bmatrix} \quad (5.45)$$

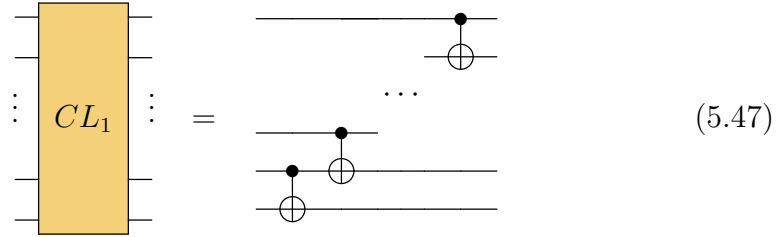
on each qubit. \square

We now introduce a new, more specialized class of Dicke states with $\nu = 2$, subject to a nearest-neighbour constraint; that is, the two $|1\rangle$ excitations must be adjacent to each other.

Definition 5.2.4. The $|D_{2\text{NN}}^n\rangle$ state is defined as:

$$\begin{aligned} |D_{2\text{NN}}^n\rangle &= \frac{1}{\sqrt{n-1}} \sum_{\ell=0}^{n-2} |2^\ell + 2^{\ell+1}\rangle \\ &= \frac{1}{\sqrt{n-1}} \sum_{\ell=0}^{n-2} |\underbrace{0 \dots 0}_\ell 11 \underbrace{0 \dots 0}_{n-2-\ell}\rangle \end{aligned} \quad (5.46)$$

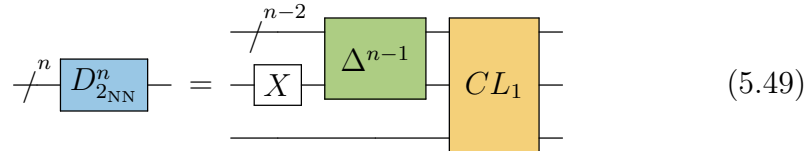
Lemma 5.2.2. The $|D_{2\text{NN}}^n\rangle$ state can be constructed by first preparing the $|D_1^{n-1}\rangle$ state and then applying a CNOT-ladder, denoted by CL_1 , defined as



thus, the final circuit will be:

$$|0^{\otimes n}\rangle \xrightarrow{D_{2\text{NN}}^n} |D_{2\text{NN}}^n\rangle \quad (5.48)$$

Where:



Proof. We have that each CNOT gate in the ladder is activated by exactly one component of the state vector:

$$|2^\ell\rangle \xrightarrow{CL_1} |2^\ell + 2^{\ell+1}\rangle, \quad (5.50)$$

so that, by linearity, the entire superposition evolves into the correct target

state:

$$\begin{aligned}
|D_1^{n-1}\rangle |0\rangle &= \frac{1}{\sqrt{n-1}} \sum_{\ell=0}^{n-2} |2^\ell\rangle \\
&\xrightarrow{CL_1} \frac{1}{\sqrt{n-1}} \sum_{\ell=0}^{n-2} |2^\ell + 2^{\ell+1}\rangle \\
&= |D_{2_{NN}}^n\rangle
\end{aligned} \tag{5.51}$$

□

Definition 5.2.5. The Dicke state with $\nu = 2$ but with k -th nearest neighbour constraint is defined as:

$$|D_{2_{kN}}^n\rangle = \frac{1}{\sqrt{n-k}} \sum_{\ell=0}^{n-k-1} |2^\ell + 2^{\ell+k}\rangle . \tag{5.52}$$

Lemma 5.2.2 can then be generalized by considering the subroutines Δ^{n-k} and CL_k , which consists of $n-k$ CNOTs of length k , with final circuit:

$$|0^{\otimes n}\rangle \xrightarrow{D_{2_{kN}}^n} |D_{2_{kN}}^n\rangle \tag{5.53}$$

where:

$$\begin{array}{c}
\text{---}^n \text{---} \boxed{D_{2_{kN}}^n} \text{---} \\
= \\
\begin{array}{c}
\text{---}^{n-k-1} \text{---} \boxed{\Delta^{n-k}} \text{---} \\
\text{---}^k \text{---} \boxed{X} \text{---} \\
\text{---} \text{---} \boxed{CL_k} \text{---}
\end{array}
\end{array} \tag{5.54}$$

Finally, we introduce the *double* variant of Dicke states by entangling them with a second register of n qubits.

Definition 5.2.6. The double Dicke state with $\nu = 1$ is defined as:

$$|\mathcal{D}_1^n\rangle = \frac{1}{\sqrt{n}} \sum_{\ell=0}^{n-1} |2^\ell\rangle |2^\ell\rangle . \tag{5.55}$$

so that:

$$|0^{\otimes 2n}\rangle \xrightarrow{\mathcal{D}_1^n} |\mathcal{D}_1^n\rangle \tag{5.56}$$

This state can be constructed by first preparing the single-excitation Dicke state $|D_1^n\rangle$ on the first register of n qubits, and then entangling it with the second register via n CNOTs, each acting between corresponding qubits of the two registers:

$$\text{---}^{2n}\text{---}\mathbb{D}_1^n\text{---} = \begin{array}{c} \text{---}^{n-1}\text{---} \\ | \\ \text{---}^n\text{---} \end{array} \begin{array}{c} \boxed{X} \\ | \\ \end{array} \begin{array}{c} \boxed{\Delta^n} \\ | \\ \end{array} \boxed{EC} \text{---} \quad (5.57)$$

where EC stands for *element-wise* CNOT and is defined as :

$$\begin{array}{c} \vdots \\ | \\ \vdots \end{array} \boxed{EC} \begin{array}{c} \vdots \\ | \\ \vdots \end{array} = \begin{array}{c} \bullet \\ | \\ \bullet \\ | \\ \vdots \\ | \\ \bullet \\ | \\ \oplus \\ | \\ \vdots \\ | \\ \vdots \\ | \\ \oplus \\ | \\ \vdots \\ | \\ \oplus \end{array} \quad (5.58)$$

Note that the n CNOTs in EC can be performed in parallel.

Definition 5.2.7. The double version of the $|D_{2_{kN}}^n\rangle$ states is defined as:

$$|D_{2_{kN}}^n\rangle = \frac{1}{\sqrt{n-k}} \sum_{\ell=0}^{n-k-1} |2^\ell + 2^{\ell+k}\rangle |2^\ell + 2^{\ell+k}\rangle, \quad (5.59)$$

The state $|D_{2_{NN}}^n\rangle$ is the special case $k = 1$.

Equation (5.59) can be prepared with the same strategy as eq. (5.56) by appending an EC entangling gate:

$$|0^{\otimes 2n}\rangle \text{---} \mathbb{D}_{2_{kN}}^n \text{---} |D_{2_{kN}}^n\rangle \quad (5.60)$$

where:

$$\text{---}^{2n}\text{---}\mathbb{D}_{2_{kN}}^n\text{---} = \begin{array}{c} \text{---}^{n-k-1}\text{---} \\ | \\ \text{---}^k\text{---} \\ | \\ \text{---}^n\text{---} \end{array} \begin{array}{c} \boxed{X} \\ | \\ \end{array} \begin{array}{c} \boxed{\Delta^{n-k}} \\ | \\ \end{array} \boxed{CL_k} \boxed{EC} \text{---} \quad (5.61)$$

To conclude this section, we evaluate the computational cost of the preparation circuits discussed above, measured in terms of the number of CNOT gates. A summary of these results is provided in table 5.2.

Dicke		Double Dicke	
State	# CNOTs	State	# CNOTs
D_1^n	$2n - 2$	\mathcal{D}_1^n	$3n - 2$
$D_{2\text{NN}}^n$	$3n - 5$	$\mathcal{D}_{2\text{NN}}^n$	$4n - 5$
$D_{2k\text{N}}^n$	$3n - 3k - 2$	$\mathcal{D}_{2k\text{N}}^n$	$4n - 3k - 2$

Table 5.2: CNOT gate count for every Dicke state subroutine

5.3 Applications on spin models

We begin by considering the Heisenberg model of quantum spins arranged on a one-dimensional chain. The spin chain consists of n sites, where each site hosts a spin- $\frac{1}{2}$ particle (such as an electron). A spin- $\frac{1}{2}$ particle is a two-level quantum system, meaning its spin degree of freedom can be described by a qubit with basis states $|0\rangle$ and $|1\rangle$, corresponding to spin up and spin down along a chosen axis. Each electron can be in a spin-up or spin-down state, and therefore its quantum state is described by a linear combination $a|\uparrow\rangle + b|\downarrow\rangle$, spanning a two-dimensional local Hilbert space. When we consider N such particles, the total Hilbert space in which the physical states reside is given by the tensor product of the local spaces, namely $\mathbb{H} = (\mathbb{C}^2)^{\otimes n}$.

Definition 5.3.1. The one-dimensional Heisenberg Hamiltonian with open boundary conditions is defined as

$$\begin{aligned}
 H = & \sum_{\ell=0}^{n-1} g^x X_\ell + g^z Z_\ell + g^y Y_\ell \\
 & + \sum_{\ell=0}^{n-2} J^x X_\ell X_{\ell+1} + J^z Z_\ell Z_{\ell+1} + J^y Y_\ell Y_{\ell+1} , \quad (5.62)
 \end{aligned}$$

where $X_\ell = \underbrace{I \otimes \cdots \otimes I}_\ell \otimes X \otimes \underbrace{I \otimes \cdots \otimes I}_{n-1-\ell}$.

The constants J^x , J^y , and J^z represent the coupling strengths along the respective spin directions. In the isotropic case, where $J_x = J_y = J_z = J$, the model is called the XXX Heisenberg model. If only two components are

Operator	$\tilde{\alpha}_{ij}$	$ i\rangle j\rangle$
$g^x X_\ell$	g^x	$ 2^\ell\rangle 0^{\otimes n}\rangle$
$g^z Z_\ell$	g^z	$ 0^{\otimes n}\rangle 2^\ell\rangle$
$g^y Y_\ell$	$-ig^y$	$ 2^\ell\rangle 2^\ell\rangle$
$J^x X_\ell X_{\ell+1}$	J^x	$ 2^\ell + 2^{\ell+1}\rangle 0^{\otimes n}\rangle$
$J^z Z_\ell Z_{\ell+1}$	J^z	$ 0^{\otimes n}\rangle 2^\ell + 2^{\ell+1}\rangle$
$J^y Y_\ell Y_{\ell+1}$	$-J^y$	$ 2^\ell + 2^{\ell+1}\rangle 2^\ell + 2^{\ell+1}\rangle$

Table 5.3: Mapping between every Pauli string in eq. (5.62) to the coefficient and ancilla in the formalism from eq. (5.16) and eq. (5.17).

equal, e.g., $J_x = J_y \neq J_z$, it is referred to as the XXZ model. The Heisenberg model captures essential features of quantum magnetism and plays a central role in the study of condensed matter systems, particularly in the context of quantum phase transitions and entanglement.

The Hamiltonian in eq. (5.62) is already expressed as a linear combination of Pauli strings, as in eq. (5.11). To construct the check-matrix formalism introduced in eq. (5.16), and to implement the corresponding P_R and P_L oracles, we map each term in eq. (5.62) to a coefficient-state pair of the form $(\tilde{\alpha}_{ij}, |i\rangle |j\rangle)$. Here, the register $|i\rangle$ encodes the support of the X and Y components, while $|j\rangle$ encodes the support of the Z components, enabling the implementation of the SELECT operator as described previously.

Example 5.3.1. The term $g^x X_\ell$ is mapped to $(g^x, |2^\ell\rangle |0\rangle)$:

$$\begin{aligned}
g^x X_\ell &= \underbrace{I \otimes \cdots \otimes I}_\ell \otimes (g^x X) \otimes \underbrace{I \otimes \cdots \otimes I}_{n-1-\ell} \\
&\rightarrow (g^x, |\underbrace{0 \dots 0}_\ell \underbrace{1 \dots 0}_{n-1-\ell}\rangle \otimes |0^{\otimes n}\rangle) \\
&= (g^x, |2^\ell\rangle |0^{\otimes n}\rangle) .
\end{aligned} \tag{5.63}$$

Table 5.3 summarizes the mapping for all terms in eq. (5.62). As a consequence, the oracle P_R must prepare the following complex state on the $2n$

ancillae:

$$\begin{aligned} \frac{1}{\sqrt{N}} \left[\sum_{\ell=0}^{n-1} \sqrt{g^x} |2^\ell\rangle |0^{\otimes n}\rangle + \sqrt{g^z} |0^{\otimes n}\rangle |2^\ell\rangle + \sqrt{-ig^y} |2^\ell\rangle |2^\ell\rangle \right. \\ \left. + \sum_{\ell=0}^{n-2} \sqrt{J^x} |2^\ell + 2^{\ell+1}\rangle |0^{\otimes n}\rangle + \sqrt{J^z} |0^{\otimes n}\rangle |2^\ell + 2^{\ell+1}\rangle \right. \\ \left. + \sqrt{-J^y} |2^\ell + 2^{\ell+1}\rangle |2^\ell + 2^{\ell+1}\rangle \right], \end{aligned} \quad (5.64)$$

where N is the same normalization factor as LCU:

$$N = n (|g^x| + |g^y| + |g^z|) + (n-1) (|J^x| + |J^y| + |J^z|) \quad (5.65)$$

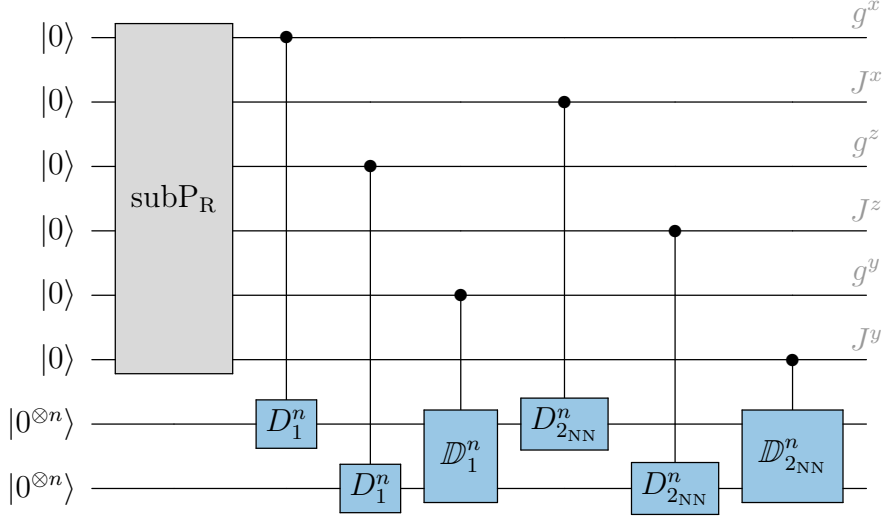
Analogously, P_L prepares the same state but with complex conjugate coefficients.

Remark 5.3.1. Grouping the one-body terms in eq. (5.64) by equal coefficients reveals that each group corresponds, up to normalization, to a Dicke state. More specifically, eq. (5.64) can be reformulated as:

$$\begin{aligned} \frac{1}{\sqrt{N}} \left(\sqrt{ng^x} |D_1^n\rangle |0^{\otimes n}\rangle + \sqrt{ng^z} |0^{\otimes n}\rangle |D_1^n\rangle + \sqrt{-ing^y} |\mathcal{D}_1^n\rangle \right. \\ \left. + \sqrt{(n-1)J^x} |D_{2_{\text{NN}}}^n\rangle |0^{\otimes n}\rangle + \sqrt{(n-1)J^z} |0^{\otimes n}\rangle |D_{2_{\text{NN}}}^n\rangle \right. \\ \left. + \sqrt{-(n-1)J^y} |\mathcal{D}_{2_{\text{NN}}}^n\rangle \right), \end{aligned} \quad (5.66)$$

where each of the states $|D_1^n\rangle$, $|\mathcal{D}_1^n\rangle$, $|D_{2_{\text{NN}}}^n\rangle$ and $|\mathcal{D}_{2_{\text{NN}}}^n\rangle$ can be implemented with linear gate count, as shown in the previous section.

In order to construct efficient circuits for P_R and P_L , we augment the system with six ancilla qubits. Since Dicke states admit efficient implementations, our approach is to design a subroutine, denoted $\text{sub}P_R$, which prepares the desired coefficient amplitudes by generating an unbalanced Dicke state.

Figure 5.3: Circuit used to implement P_R for the Heisenberg model.

Definition 5.3.2. The subroutine subP_R prepares the following state:

$$\begin{aligned} \frac{1}{\sqrt{N}} & \left(\sqrt{ng^x} |100000\rangle + \sqrt{(n-1)J^x} |010000\rangle \right. \\ & + \sqrt{ng^z} |001000\rangle + \sqrt{(n-1)J^z} |000100\rangle \\ & \left. + \sqrt{-ing^y} |000010\rangle + \sqrt{-(n-1)J^y} |000001\rangle \right), \end{aligned} \quad (5.67)$$

it can be implemented by a circuit as shown in eq. (5.38).

Lemma 5.3.1. The circuit in fig. 5.3 prepares the state:

$$\begin{aligned} \frac{1}{\sqrt{N}} & \left(\sqrt{ng^x} |100000\rangle |D_1^n\rangle |0^{\otimes n}\rangle + \sqrt{ng^z} |001000\rangle |0^{\otimes n}\rangle |D_1^n\rangle + \sqrt{-ing^y} |000010\rangle |D_1^n\rangle \right. \\ & + \sqrt{(n-1)J^x} |010000\rangle |D_{2NN}^n\rangle |0^{\otimes n}\rangle + \sqrt{(n-1)J^z} |000100\rangle |0^{\otimes n}\rangle |D_{2NN}^n\rangle \\ & \left. + \sqrt{-(n-1)J^y} |000001\rangle |D_{2NN}^n\rangle \right), \end{aligned} \quad (5.68)$$

Proof. We begin with the initial state

$$|0^{\otimes 6}\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle.$$

After applying the subP_R circuit, the system is prepared in the following

superposition:

$$\begin{aligned} & \frac{1}{\sqrt{N}} \left(\sqrt{ng^x} |100000\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle + \sqrt{(n-1)J^x} |010000\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle \right. \\ & \quad + \sqrt{ng^z} |001000\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle + \sqrt{(n-1)J^z} |000100\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle \\ & \quad \left. + \sqrt{-ing^y} |000010\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle + \sqrt{-(n-1)J^y} |000001\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle \right). \end{aligned}$$

Next, we sequentially apply the controlled Dicke operations. Each control qubit is associated with exactly one Dicke state gate, and only one of them is active for each term in the superposition. For example, after applying the first controlled Dicke gate (conditioned on the first control qubit), we obtain:

$$\begin{aligned} & \frac{1}{\sqrt{N}} \left(\sqrt{ng^x} |100000\rangle |D_1^n\rangle |0^{\otimes n}\rangle + \sqrt{(n-1)J^x} |010000\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle \right. \\ & \quad + \sqrt{ng^z} |001000\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle + \sqrt{(n-1)J^z} |000100\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle \\ & \quad \left. + \sqrt{-ing^y} |000010\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle + \sqrt{-(n-1)J^y} |000001\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle \right). \end{aligned}$$

Proceeding similarly, each controlled Dicke operation modifies only the corresponding term in the superposition, preparing either D_1^n , D_{2NN}^n , \mathbb{D}_1^n and \mathbb{D}_{2NN}^n . At the end of this process, the full state matches the expression in equation (5.68), as claimed. \square

Remark 5.3.2. P_L is constructed in an analogous way by considering the complex conjugate of the coefficients in the sub P_R .

Proposition 5.3.1. The circuit $U_H = P_R \cdot \text{SELECT} \cdot P_L^\dagger$ is a $(2n+6, N, 0)$ -block encoding of the Heisenberg Hamiltonian defined in eq. (5.62).

Proof. As shown in lemma 5.3.1, the application of the P_R circuit prepares the state in (5.64), tensored with an arbitrary input state $|\psi\rangle$. This state can

be written explicitly as:

$$\begin{aligned} & \frac{1}{\sqrt{N}} \left[\sum_{\ell=0}^{n-1} \sqrt{g^x} |100000\rangle |2^\ell\rangle |0^{\otimes n}\rangle + \sqrt{g^z} |001000\rangle |0^{\otimes n}\rangle |2^\ell\rangle \right. \\ & \quad + \sqrt{-ig^y} |000010\rangle |2^\ell\rangle |2^\ell\rangle \\ & \quad + \sum_{\ell=0}^{n-2} \sqrt{J^x} |010000\rangle |2^\ell + 2^{\ell+1}\rangle |0^{\otimes n}\rangle + \sqrt{J^z} |000100\rangle |0^{\otimes n}\rangle |2^\ell + 2^{\ell+1}\rangle \\ & \quad \left. + \sqrt{-J^y} |000001\rangle |2^\ell + 2^{\ell+1}\rangle |2^\ell + 2^{\ell+1}\rangle \right] \otimes |\psi\rangle, \end{aligned}$$

where we used the definition of Dicke states to cancel the \sqrt{n} and $\sqrt{n-1}$ factors. Next, we apply the SELECT operator. This applies a Pauli operator to $|\psi\rangle$ depending on the computational basis states in the ancillary registers. The resulting state is:

$$\begin{aligned} & \frac{1}{\sqrt{N}} \left[\sum_{\ell=0}^{n-1} \sqrt{g^x} |100000\rangle |2^\ell\rangle |0^{\otimes n}\rangle X_\ell |\psi\rangle \right. \\ & \quad + \sqrt{g^z} |001000\rangle |0^{\otimes n}\rangle |2^\ell\rangle Z_\ell |\psi\rangle \\ & \quad + \sqrt{-ig^y} |000010\rangle |2^\ell\rangle |2^\ell\rangle Z_\ell X_\ell |\psi\rangle \\ & \quad + \sum_{\ell=0}^{n-2} \sqrt{J^x} |010000\rangle |2^\ell + 2^{\ell+1}\rangle |0^{\otimes n}\rangle X_\ell X_{\ell+1} |\psi\rangle \\ & \quad + \sqrt{J^z} |000100\rangle |0^{\otimes n}\rangle |2^\ell + 2^{\ell+1}\rangle Z_\ell Z_{\ell+1} |\psi\rangle \\ & \quad \left. + \sqrt{-J^y} |000001\rangle |2^\ell + 2^{\ell+1}\rangle |2^\ell + 2^{\ell+1}\rangle Z_\ell X_\ell Z_{\ell+1} X_{\ell+1} |\psi\rangle \right]. \end{aligned}$$

We now apply P_L to the state $|0^{\otimes 6}\rangle |0^{\otimes n}\rangle |0^{\otimes n}\rangle$. This gives:

$$\begin{aligned} & \frac{1}{\sqrt{N}} \left[\sum_{\ell=0}^{n-1} \sqrt{g^{x*}} |100000\rangle |2^\ell\rangle |0^{\otimes n}\rangle + \sqrt{g^{z*}} |001000\rangle |0^{\otimes n}\rangle |2^\ell\rangle \right. \\ & \quad + \sqrt{-ig^{y*}} |000010\rangle |2^\ell\rangle |2^\ell\rangle \\ & \quad + \sum_{\ell=0}^{n-2} \sqrt{J^{x*}} |010000\rangle |2^\ell + 2^{\ell+1}\rangle |0^{\otimes n}\rangle + \sqrt{J^{z*}} |000100\rangle |0^{\otimes n}\rangle |2^\ell + 2^{\ell+1}\rangle \\ & \quad \left. + \sqrt{-J^{y*}} |000001\rangle |2^\ell + 2^{\ell+1}\rangle |2^\ell + 2^{\ell+1}\rangle \right]. \end{aligned}$$

$$\langle 0^{\otimes 6} | \langle 0^{\otimes n} | \langle 0^{\otimes n} | P_L^\dagger \cdot \text{SELECT} \cdot P_R | 0^{\otimes 6} \rangle | 0^{\otimes n} \rangle | 0^{\otimes n} \rangle | \psi \rangle .$$
$$\frac{1}{N} \left[\sum_{\ell=0}^{n-1} g^x X_{\ell} |\psi\rangle + g^z Z_{\ell} |\psi\rangle - i g^y Z_{\ell} X_{\ell} |\psi\rangle \right. \\ \left. + \sum_{\ell=0}^{n-2} J^x X_{\ell} X_{\ell+1} |\psi\rangle + J^z Z_{\ell} Z_{\ell+1} |\psi\rangle - J^y Z_{\ell} X_{\ell} Z_{\ell+1} X_{\ell+1} |\psi\rangle \right].$$

5.4 Compression of P_R for the Heisenberg Model

Lemma 5.4.1. If the initial state is $|0^{\otimes n}\rangle$, the controlled gate D_1^n is equivalent to controlling only the initial X gate while the subroutine Δ^n does not need to be controlled:

Proof. Let apply the circuit to the state $(a|0\rangle + b|1\rangle)|0^{\otimes n}\rangle$:

since $\Delta^n |0^{\otimes n}\rangle = |0^{\otimes n}\rangle$.

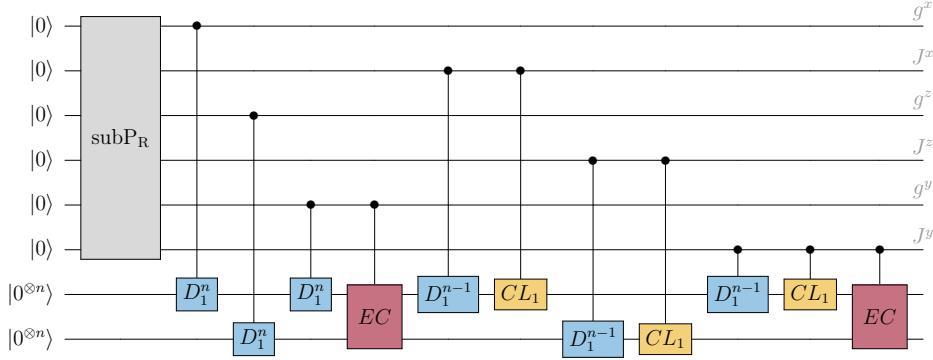


Figure 5.4: Decomposition of the gates in fig. 5.3.

Proof. If we apply the left circuit to the state $(a|01\rangle + b|10\rangle)|0^{\otimes n}\rangle$, we get the state $(a|01\rangle + b|10\rangle)C|0^{\otimes n}\rangle$. We get the same state even if we apply the right circuit we get:

$$\begin{aligned}
 (a|01\rangle + b|10\rangle)|0^{\otimes n}\rangle &\xrightarrow{\text{CNOT}_1} a|01\rangle|0^{\otimes n}\rangle + b|11\rangle|0^{\otimes n}\rangle \\
 &\xrightarrow{\text{controlled-}C} a|01\rangle C|0^{\otimes n}\rangle + b|11\rangle C|0^{\otimes n}\rangle \\
 &\xrightarrow{\text{CNOT}_2} a|01\rangle C|0^{\otimes n}\rangle + b|10\rangle C|0^{\otimes n}\rangle \quad (5.75)
 \end{aligned}$$

□

This lemma is particularly useful in simplifying subroutines that apply the same gate multiple times to a common target, but with different control qubits. In such cases, when the control register is restricted to the single-excitation subspace (i.e., an unbalanced Dicke state), the number of controlled operations can be reduced. Notably, this simplification is employed in the implementation of the CL_k subroutine, which appears in both the $D_{2_{kN}}^n$ and $\mathbb{D}_{2_{kN}}^n$ state preparation protocols. It is also used in the construction of the EC subroutine, which is responsible for duplicating a Dicke state efficiently. We proceed with the compression of the circuit presented in fig. 5.3. We begin by decomposing the Dicke state oracles $D_{2_{NN}}^n$, \mathbb{D}_1^n , and $\mathbb{D}_{2_{NN}}^n$. Each oracle is split into two parts: the preparation of the state $|D_1^n\rangle$, and either a CL_1 , a EC , or both, depending on the specific oracle. The resulting circuit is shown in fig. 5.4. Since the $|\text{subP}_R\rangle$ state on the six additional ancillae is an unbalanced Dicke state with a single excitation, the gates in fig. 5.4

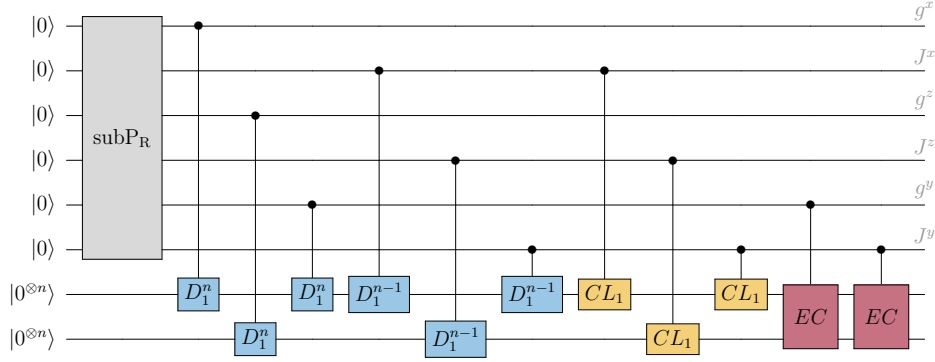


Figure 5.5: Commutations of the controlled gates of fig. 5.4.

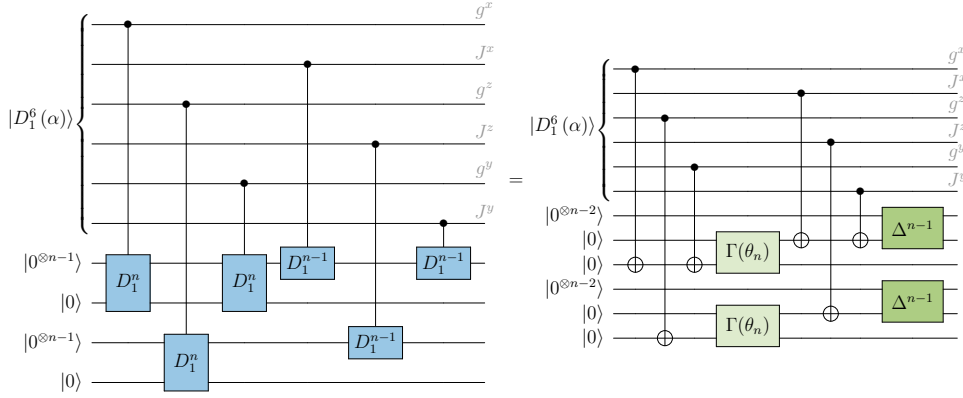


Figure 5.6: Compression of controlled Dicke gates.

can be rearranged as shown in fig. 5.5. In this new configuration, we can apply the compression techniques from lemma 5.4.3 to reduce the number of CL_1 and EC operations, respectively. Furthermore, the portion of the circuit responsible for preparing balanced Dicke states can also be simplified by combining lemma 5.4.1 and lemma 5.4.2 as shown in fig. 5.6. This sequence of optimizations leads to the final, compressed implementation of the P_R circuit for the Heisenberg model shown in fig. 5.7.

Finally, we estimate the computational cost of implementing the FOQCS-LCU algorithm for the Heisenberg model. We then compare its performance with other block-encoding techniques, such as LCU and FABLE. As a resource metric, we consider the number of CNOTs required in the respective circuits. Specifically, we first report the number of Toffoli gates and two-qubit gates separately. At the final stage, we decompose these gates into

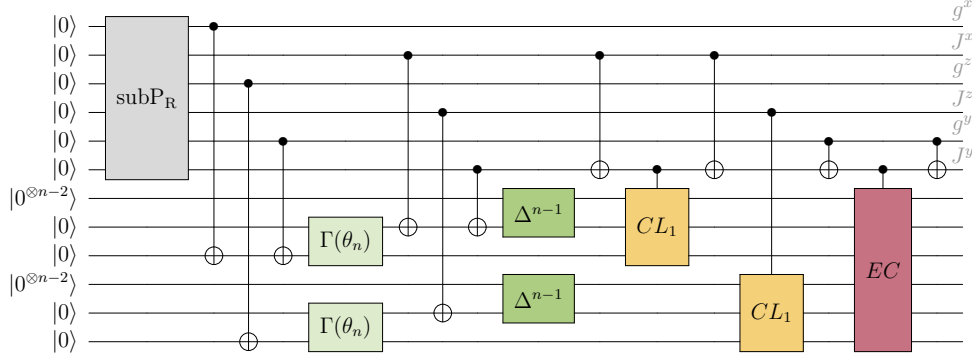


Figure 5.7: Compact implementation of P_R for the Heisenberg model

CNOTs to obtain the total CNOT count.

Theorem 5.4.1. The FOQCS-LCU block encoding of the Heisenberg Hamiltonian with n sites requires $6n - 4$ Toffoli gates and $10n + 32$ two-qubit gates, resulting in a total of $46n + 8$ CNOT gates.

Proof. We analyze the circuit in fig. 5.7 that implements the oracle P_R (with P_L exhibiting exactly the same complexity). This circuit is composed of the following components:

- **The $\text{sub}P_R$ subroutine (on 6 qubits):** this subroutine contains 5 $\Gamma(\theta)$. We show in lemma 5.2.1 how to construct each $\Gamma(\theta)$ using 2 CNOTs, yielding a total of 10 CNOTs.
- **Two $\Gamma(\theta_n)$ subroutines:** for a total of 4 CNOTs.
- **Two Δ^{n-1} subroutines:** each Δ^{n-1} consists of $n - 2$ subroutines $\Gamma(\theta)$. We need then $2(n - 2)$ CNOTs for each Δ^{n-1} , resulting in a total of $4(n - 2)$ CNOTs.
- **Two controlled CL_1 gates:** each CL_1 involves $n - 1$ CNOTs. When controlled, these gates become $n - 1$ Toffoli gates each, totaling $2(n - 1)$ Toffolis.
- **One controlled EC :** this gate includes n CNOTs, and when controlled, these become n Toffolis.

- **Ten additional CNOT gates.**

In total, the subroutine P_R requires then $3n - 2$ Toffoli gates and $4n + 16$ CNOTs (and the same applies to P_L).

The SELECT part consists only of $2n$ CNOTs, since cZ is implemented with one CNOT and two Hadamard gates. Thus, the total number of Toffoli gates is $6n - 4$ and the total number of CNOTs is $10n + 32$ for the FOQCS-LCU block encoding of the Heisenberg Hamiltonian. If we further decompose the Toffoli gates, each Toffoli would require 6 CNOTs so that the overall cost in terms of CNOTs becomes:

$$6 \cdot (6n - 4) + (10n + 32) = 46n + 8. \quad (5.76)$$

□

We finally compare our approach with the standard LCU method and with the Fast Approximate Block Encodings (FABLE) algorithm [10], evaluated at two different accuracy thresholds. All circuits were generated using the `qiskit` software [28]. It is an open-source quantum computing framework developed by IBM, which provides tools for designing, simulating, and executing quantum circuits on both simulators and real quantum hardware. For standard LCU, we implemented the circuit shown in fig. 5.1, where the P_R and P_L subroutines, defined in eq. (5.2) and eq. (5.3) respectively, were instantiated through `qiskit`'s default state initialization routine [29]. To further refine the resulting circuits, we applied post-compilation optimization with the Berkeley Quantum Synthesis Toolkit (BQSKit) [30], which performs state-of-the-art circuit depth reduction and gate cancellation. BQSKit was applied to all data points for the Heisenberg model. BQSKit provides only modest improvements, as the generic LCU implementation does not exploit the algebraic or physical structure of the Hamiltonian. As a result, it exhibits poorer scaling compared to our FOQCS-LCU method, which explicitly leverages problem structure in its design. In fact, the LCU algorithm applied to the Heisenberg Hamiltonian has a gate complexity of $\mathcal{O}(n \log^\beta n)$, while our

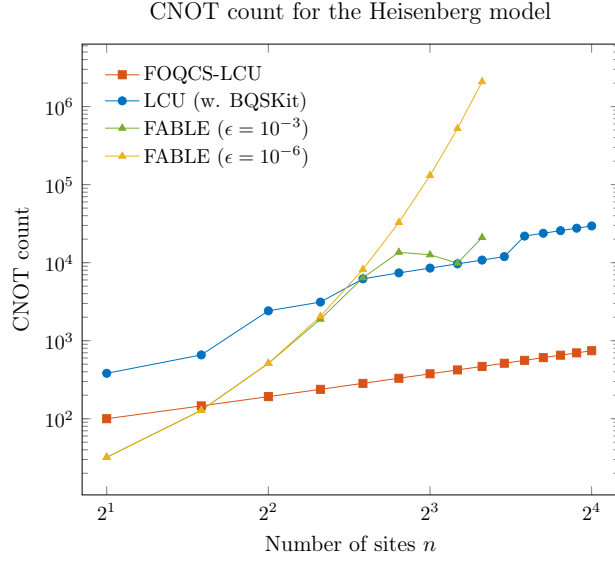


Figure 5.8: CNOT count from FOQCS-LCU, LCU, and FABLE for the Heisenberg model, with FABLE evaluated at precisions $\epsilon = 10^{-3}$ and $\epsilon = 10^{-6}$.

Fast One-Qubit Control Select LCU approach achieves a strictly linear scaling of $\mathcal{O}(n)$ CNOT gates by fully exploiting the underlying structure of the problem.

In conclusion, our FOQCS-LCU implementation achieves more than an order-of-magnitude improvement in the CNOT count relative to the standard and structure-agnostic LCU. This gain arises from two key advantages: (i) the replacement of multi-controlled operations in the SELECT subroutine of fig. 5.1 with singly-controlled X and Z gates, as shown in fig. 5.2, and (ii) the use of structured state preparation based on Dicke states.

Chapter 6

Conclusion

In the final chapter, we presented the main contribution of this thesis: a novel block encoding framework, denoted Fast One-Qubit Control Select LCU (FOQCS-LCU), which addresses a key limitation of the standard LCU approach—namely, the reliance on deep and costly multi-controlled operations in the SELECT oracle. In our formulation, the SELECT operator is implemented using only n CNOT and n cZ gates, significantly reducing circuit depth and complexity. We then introduced an efficient method for preparing Dicke states—quantum superpositions of basis states with a fixed number of qubits in the $|1\rangle$ state—which are used to optimize the state preparation and uncomputation oracles. To demonstrate the practical utility of our method, we constructed an explicit FOQCS-LCU circuit for the Heisenberg Hamiltonian and proposed a circuit compression strategy that exploits gate structure to further reduce the CNOT count. The CNOT gate is a standard two-qubit entangling gate that is both more resource-intensive and more error-prone than single-qubit gates on most quantum hardware. As a result, the number of CNOT gates in a quantum circuit serves as a practical and widely accepted measure of its overall complexity and feasibility for near-term implementation. This compression was achieved by identifying redundant applications of the same gates and replacing them with a single modified version, exploiting the structure of the matrix and the one-to-one

mapping between basis states and tensor products of Pauli operators.

Finally, we have demonstrated through numerical benchmarks that our method reduces the total CNOT gate count by an order of magnitude compared to existing LCU-based techniques. This result is particularly encouraging, as the reduced cost suggests the possibility of implementing this algorithm on early fault tolerant quantum devices. Moreover, this work has laid the foundation for studying other models with similar structure. Looking ahead, a particularly promising direction is the application of this framework to fermionic systems, such as the Fermi-Hubbard model [\[31\]](#), where exploiting structure could similarly lead to efficient quantum circuit implementations.

Bibliography

- [1] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.
- [2] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6/7):467–488, 1982.
- [3] 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*, STOC '19, page 193–204. ACM, June 2019.
- [4] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2(4), December 2021.
- [5] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), October 2009.
- [6] I.M. Georgescu, S. Ashhab, and Franco Nori. Quantum simulation. *Reviews of Modern Physics*, 86(1):153–185, March 2014.
- [7] Bruce M. Boghosian and Washington Taylor. Simulating quantum mechanics on a quantum computer. *Physica D: Nonlinear Phenomena*, 120(1–2):30–42, September 1998.

- [8] Guang Hao Low and Isaac L. Chuang. Optimal hamiltonian simulation by quantum signal processing. *Physical Review Letters*, 118(1), January 2017.
- [9] Guang Hao Low and Isaac L. Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, July 2019.
- [10] Daan Camps and Roel Van Beeumen. Fable: Fast approximate quantum circuits for block-encodings. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, page 104–113. IEEE, September 2022.
- [11] Daan Camps, Lin Lin, Roel Van Beeumen, and Chao Yang. Explicit quantum circuits for block encodings of certain sparse matrices, 2023.
- [12] R. H. Dicke. Coherence in spontaneous radiation processes. *Phys. Rev.*, 93:99–110, Jan 1954.
- [13] Ronald de Wolf. Quantum computing: Lecture notes, 2023. <https://arxiv.org/abs/1907.09415>.
- [14] Lin Lin. Lecture notes on quantum algorithms for scientific computation, 2022. <https://arxiv.org/abs/2201.08309>.
- [15] W. Scherer. *Mathematics of Quantum Computing: An Introduction*. Springer International Publishing, 2019.
- [16] J. Ossorio-Castillo and José M. Tornero. Quantum computing from a mathematical perspective: a description of the quantum circuit model, 2025. <https://arxiv.org/abs/1810.08277>.
- [17] Guang Hao Low, Theodore J. Yoder, and Isaac L. Chuang. Methodology of resonant equiangular composite quantum gates. *Physical Review X*, 6(4), December 2016.
- [18] Daan Camps and Roel Van Beeumen. Approximate quantum circuit synthesis using block encodings. *Phys. Rev. A*, 102:052411, Nov 2020.

- [19] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, November 1995.
- [20] Mikko Möttönen, Juha J. Vartiainen, Ville Bergholm, and Martti M. Salomaa. Quantum circuits for general multiqubit gates. *Physical Review Letters*, 93(13), September 2004.
- [21] Fino and Algazi. Unified matrix treatment of the fast walsh-hadamard transform. *IEEE Transactions on Computers*, C-25(11):1142–1146, 1976.
- [22] Christoph Sünderhauf, Earl Campbell, and Joan Camps. Block-encoding structured matrices for data input in quantum computing. *Quantum*, 8:1226, January 2024.
- [23] Martina Nibbi and Christian B. Mendl. Block encoding of matrix product operators. *Physical Review A*, 110(4), October 2024.
- [24] Emanuel Malvetti, Raban Iten, and Roger Colbeck. Quantum circuits for sparse isometries. *Quantum*, 5:412, March 2021.
- [25] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. Quantum state preparation with optimal circuit depth: Implementations and applications. *Phys. Rev. Lett.*, 129:230504, Nov 2022.
- [26] Lvzhou Li and Jingquan Luo. Nearly optimal circuit size for sparse quantum state preparation, 2025.
- [27] Andreas Bartschi and Stephan Eidenbenz. *Deterministic Preparation of Dicke States*, page 126–139. Springer International Publishing, 2019.
- [28] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation,

- Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, 2024.
- [29] Raban Iten, Roger Colbeck, Ivan Kukuljan, Jonathan Home, and Matthias Christandl. Quantum circuits for isometries. *Phys. Rev. A*, 93:032318, Mar 2016.
- [30] Ed Younis, Costin C Iancu, Wim Lavrijsen, Marc Davis, Ethan Smith, and USDOE. Berkeley quantum synthesis toolkit (bqskit) v1, 04 2021.
- [31] Wikipedia contributors. Hubbard model — wikipedia, the free encyclopedia, 2025. https://en.wikipedia.org/wiki/Hubbard_model, [Accessed : 2025 – 07 – 12].