# Quantum Computing

Vincenzo Savona

Center for Quantum Science and Engineering, EPFL

December 25, 2024

# Acknowledgements

I am deeply indebted to Khurshed Porus Fitter, who devoted his talent, time, and energy to the deeply critical writing of these lecture notes, starting from a very crude draft and my handwritten notes. Most of what is clear and insightful in this document is due to him.

# Contents

# Chapter 1

# Introduction

## What is Quantum Computing?

In October 2019, Google announced they reached *quantum supremacy*. In short, they managed to employ a programmable and (rather) general-purpose quantum device to execute a computational task in a matter of minutes, which would take years on a conventional modern supercomputer.[1]

The goal of this course is to develop the knowledge required to fully understand the extent and implications of this ground-breaking result, as well as explore the possibilities and opportunities that quantum computing opens in all areas of science and society. Hopefully, it will also raise your awareness about the rapidly developing world of quantum computing and give you the tools to critically assess the enormous flow of news and information about progress in quantum science, to which we are exposed every day in the media.

## Where does it originate?

The idea of quantum information and quantum computing emerged at the beginning of the 80's, particularly through the visionary work of Richard Feynman. It was already known at the time that simulating the evolution of a many-body system according to the laws of quantum mechanics, using a traditional (i.e. *classical*, as opposed to *quantum*) computer, is a computationally hard task. By "hard" here, I mean that the time it takes for the best-known algorithm increases exponentially with the "size" of the system, which may be defined here as the number of particles or the degrees of freedom. Such

[1] See the original research article. IBM soon rebutted Google's estimate by arguing that a better usage of mass storage would reduce the computational time to a few days, but this is not the point. The point is that a quantum computer could outperform the biggest available classical computers on a programmable task. Later, the quantum advantage was again demonstrated on the 66-qubit Chinese quantum processor *Zuchongzhi* 2.1 by performing a computational task that is non-treatable on a modern supercomputer even when accounting for IBM's improved algorithmic solutions. Very recently, the methodology used in these two works to assess quantum supremacy was proven to be flawed. This debate, strongly rooted in the laws and predictions of quantum mechanics, shows how difficult it is even only to assess the usefulness of a quantum computer, and how the notions and concepts treated in this course will be useful to gain a critical look on this rapidly evolving field.

exponential scaling of the run time makes the problem virtually non-treatable. This was commonly seen as a limitation in our capacity to model complex quantum systems. Thinking out of the box, Richard Feynman reversed the argument and saw this as an opportunity instead. If I want to simulate a quantum system of size $N$, evolving over a time $t$, a classical computer will take some time $T_N \sim \mathrm{O}(2^N)$. If I now want to simulate a system of the same kind but twice as large, the classical computer will take a time $T_{2N} \sim \mathrm{O}((2^N)^2)$ which is exponentially larger than $T_N$. Nature, on the other hand, per definition, will always take the same time $\mathrm{O}(t)$, and therefore performs exponentially better than any classical computer in this particular computational task! Hence, in 1982, Feynman proposed the idea that a quantum system could be simulated efficiently using another appropriately engineered quantum system, on which we may have more control. This proposal marked the birth of the field of *quantum simulation*. In the same years, Feynman collaborated with engineers to study the thermodynamics of computation. One of their main conclusions was that, in order to achieve maximal thermodynamic efficiency (i.e. dissipate the least possible amount of heat), electronic logical gates should be reversible, namely, it should be possible to trace back the input from the output.[2] The state of an isolated quantum system evolves in time according to a unitary transformation (more about this later), and this evolution is naturally reversible. From there, the idea of a quantum computation paradigm based on quantum logic gates rapidly emerged.

# A brief history

Major breakthroughs came with the development of the Deutsch (1985) and the Deutsch-Jozsa quantum algorithms (1992). These algorithms showed for the first time that the quantum computation paradigm not only could work as a computational protocol but could also execute certain specific tasks with an exponential speedup with respect to any classical algorithm. In 1994, Peter Shor devised the first useful quantum algorithm, able to find prime factors of integers, with an exponential speedup compared to the best-known classical algorithms. In 1996, Lov Grover set another milestone by devising an algorithm to search unstructured databases with quadratic speedup as compared to the best classical algorithm. From Shor's and Grover's algorithms on, the field of quantum computing flourished, both theoretically and experimentally. In particular, a worldwide experimental effort was undertaken to develop technological platforms and architectures that could support quantum information.

It is very important to state at this point that the quantum computation paradigm is not universally efficient. In other words, it is (by far!) not possible to devise quantum algorithms to speed up any known computational tasks. Quantum computing is only favorable to certain specific computational problems. A wide and rapidly expanding field of research is, therefore, that of quantum software engineering, which aims at discovering new tasks for which quantum computing is efficient, and to develop and optimize the corresponding quantum algorithms.

Today's most advanced quantum computers are based on three technologies, superconducting quantum circuits, trapped ions, and Rydberg atoms, while a fourth technology employing solid-state spin qubits is emerging. All these technological solutions are affected by errors induced by the interactions of the quantum device with its environment.

[2] This is not true of conventional logical gates such as AND, OR, and XOR.

The lowest error rates at the time of writing these notes are about 0.0013 error per elementary two-qubit quantum operation and 0.0001 error per elementary one-qubit operation. John Preskill of Caltech has denoted current quantum devices as *Noisy Intermediate-Scale Quantum Hardware*, or NISQ for short. As a comparison, in modern computer hardware, the error rate is $10^{-10}$ errors per bit per hour. For most applications of conventional computers, these errors are not corrected and result in almost no significant corruption of data (most of them will likely affect one pixel of one frame of one video from your last holiday). Although error rates in quantum hardware are much higher, one goal of quantum computing is to develop a framework in which some computational tasks can still be executed in the presence of errors and still lead to substantial results. This is why today, the domain of quantum computing is broadly divided into two sub-domains: fault-tolerant and non-fault-tolerant, a.k.a. NISQ, quantum computing.

Fault-tolerant quantum computing (unlike the name suggests) assumes that errors are taken care of in some way, either through efficient hardware or by leveraging quantum error correction codes. Therefore, when writing a quantum algorithm, one can assume that the algorithm is executed by the quantum computer as is. This is the simplest case and the one that we will learn first. After having introduced the basics, we will study the most important fault-tolerant quantum algorithms, including the original version of Shor's factoring algorithm, Grover's quantum search algorithm, the digital quantum simulation algorithm (realizing Feynman's original vision), and other important algorithms[3].

Fault-tolerant quantum algorithms assume the existence of some kind of error correction scheme. In this course, we will learn the basics of how errors in a quantum computer occur, how they are modeled, and how they can be corrected. More specifically, we will introduce the stabilizer formalism and study some simple stabilizer error-correction codes, such as Shor's error correction code. Stabilizer codes use redundancy by encoding one logical qubit onto a set of $N(> 1)$ physical qubits. The code makes it possible to detect the occurrence of an error on a physical qubit through a quantum measurement of an *error syndrome* while preserving the quantum information carried by the qubit.[4] The detected error can then be corrected by applying the corresponding inverse operation. In a recent experiment on their Sycamore quantum processor, Google produced a proof of principle of exponential error suppression in a quantum memory through quantum error correction. Very recently, almost full quantum error correction protocols on a large number of physical quantum bits were demonstrated by Google, Quantinuum, and QuEra, using respectively the three mainstream quantum computing technologies.

In a real-world scenario, the processes of measuring an error syndrome and correcting the detected error are themselves affected by errors. In addition, error syndromes cannot be measured with arbitrarily high frequency. It may therefore happen that a quantum algorithm executes one or few operations onto a faulty quantum memory before the errors are detected and corrected. A legitimate question is whether we can make sure that quantum errors can still be corrected with a reasonable computational overhead, even while accounting for the fact that the error correction procedure is itself prone to errors. This possibility is called *fault tolerance*. One of the major achievements of quantum information in the last two decades is the proof that fault-tolerance is possible

[3] You have a rather comprehensive list at the Quantum Algorithm Zoo

[4] Preserving the quantum information is the nontrivial goal of these codes, as according to quantum mechanics, any measurement on a quantum system modifies its state through the collapse of the wavefunction.

within rather general assumptions, in particular under the assumption that errors occur locally on one qubit – or at most on a few neighboring qubits – independently. In the course, we will introduce the basic concepts related to fault tolerance and the challenges posed by it. We will also discuss how errors can occur in physical realizations of quantum architectures and how they can be modeled, both within the simplified digital error model and with a more accurate description using the theory of open quantum systems.

Non-fault-tolerant quantum computing assumes that errors do occur during the execution of a quantum algorithm and are not corrected. The algorithm must therefore be designed to be resilient to errors in some way. There are two strategies to do so. First, the algorithm should execute in a short number of elementary operations, i.e. be "shallow". If the error rate per operation is constant, a shallow enough algorithm may run with zero or very few errors. Second, the outcome of the computation should be something that can be statistically inferred by running the algorithm several times. Combined together, these two strategies result in *hybrid quantum algorithms*, which alternate over several cycles of (a) execution of a shallow quantum routine – which provides the actual quantum speedup – and (b) a classical routine that processes the resulting data. Hybrid algorithms conceived for current and near-future NISQ hardware hold great promise to solve many problems in science, technology and society, ranging from quantum chemistry simulations to material science, drug discovery, optimization problems in engineering, and artificial intelligence, among others, with unprecedented efficiency. Here, we will provide an overview of three of the most promising hybrid algorithms, namely the Variational Quantum Eigensolver, for the simulation of ground and excited states – and thus of the physical and chemical properties – of molecules and materials, the Quantum Approximate Optimization Algorithm, for the search of approximate solutions of complex combinatorial optimization problems, and the Variational Quantum Dynamics, to simulate the time-evolution of many-body quantum systems.

When executing NISQ quantum algorithms, it is still possible to mitigate the effects of errors on the result. The field of *quantum error mitigation* is rapidly expanding, encompassing a variety of techniques to infer the effect of errors on the estimate of specific output quantities using classical post-processing, in the attempt to extrapolate what the result would be in the absence of errors. While error mitigation techniques do not scale favorably with increasing size of the quantum computational task, they still make it possible to push the performance of NISQ hardware to remarkable levels of efficiency. An example is the recent result by IBM who achieved the digital simulation of the dynamics of an interacting system of 127 spins using error mitigation. While this result is a milestone in the quest for quantum advantage, it also spurred numerous followup studies that were able to reproduce the computational task easily on a classical computer, thanks to the use of advanced approximate techniques for the simulation of many-body quantum systems.

Time permitting, at the end of the course, we will survey the most recent developments in quantum computing.

This course will be strongly focused on the notion of digital quantum computing and will not cover topics such as analog quantum computing, linear quantum computing, measurement-based quantum computing, or quantum simulation. It will also not be a general course on quantum information, therefore not covering the topics of quantum communication and quantum sensing. Finally, as I am not an expert on the theory of classical or quantum computational complexity, I will address these notions in a minimal fashion. Those who are interested in a more in-depth account of these topics may want to read, for instance, the vast mole of work by Scott Aaronson – one of the world's leading experts on computational complexity and quantum computing.

# Chapter 2

# A crash course on Quantum Mechanics

Quantum Mechanics was developed between 1900 and 1925 as a revolutionary theory of physical reality (the other revolution being relativity, developed around the same time). It followed what is known as the *catastrophe of classical physics*. Classical physics, i.e. Newtonian mechanics and Maxwell electrodynamics, were unable to explain several phenomena and had some inconsistent physical predictions (black-body radiation, spectra of atoms, photoelectric effect, etc.)

It required 25 years to be completed because it provides a completely new paradigm for the description of the physical quantities and the laws governing time evolution. Naturally, to develop a revolutionary paradigm of physics would mean to first incorporate and nurture a completely novel way of thinking about physical phenomena. This, most of the time, is quite counter-intuitive owing to the stark differences between the quantum and classical theories of physics. In what follows, we will first motivate the need for quantum mechanics and then summarize the key postulates.

In all pre-quantum (classical) physical theories, the focus was quite naturally the mathematical description of physical *observables* (quantities we can measure in an experiment). Newtonian mechanics of a point-particle established "equations of motion" describing how the position, velocity, acceleration and further derivatives vary in time. Similarly, it describes the motion of a rotating rigid body and Maxwell's theory describes the dynamics of electric and magnetic fields (which can be measured directly using test charges and dipoles). In a theory of physical reality before the 1900s there always existed a direct mathematical description of the observables.

Sometimes there are too many observables and they generally obey very complicated and complex dynamics. In this case, physicists resorted to statistical theories that laid the probabilistic mathematical description of average thermodynalical quantities. Thermodynamics describes average macroscopic quantities like pressure, volume, temperature and energy. The Maxwell-Boltzmann distribution

$$f(v)d^3v = \left(\frac{m}{2\pi kT}\right)^{3/2} \exp\left(-\frac{mv^2}{2kT}\right)d^3v \tag{2.1}$$

is the probability distribution function (PDF) of the speed of molecules in a gas at equilibrium, at a temperature $T$, per unit volume $d^3v$ in velocity the space. It tells you what fraction of the molecules are on average moving at speed $v$ (i.e. an observable). A similar explanation can be provided for the Boltzmann equations describing the dynamics of $f(r,v)d^3rd^3v$ for a gas out of thermal equilibrium. Probabilities are naturally introduced when we decide to give up most of the microscopic information and retain only statistical averages. Nonetheless, we always work with a direct mathematical representation of the observable quantities.

In quantum mechanics, this is no longer the case. Here, the *state* of a system–that contains all the information about its properties at a given time–is described by a vector in a quite abstract and (usually) infinite-dimensional vector space! We will soon learn the essential characteristics of these vector spaces. Consequently, physical quantities or observables are described by linear operators acting on these state vectors. This leads to the natural question "*What is the link between these quantum mechanical observables and physically observable quantities?*". Or, considering a practical example, "*How do we use the quantum theory to predict the trajectory of a particle?*".

We will see that the link to observables is established in the third axiom, which is the heart of the theory. More generally, quantum mechanics is an axiomatic theory, and many efforts are continually made, even today, to lower the number of required axioms. However, a reasonable question would be "Why work with a theory that seems to be so strange and axiomatic?" Simply because, although axiomatic in nature, it is by far the most successful physical theory in the history of mankind, with thousands of successful predictions and not a single violation (with the exclusion of mechanics inside black holes, but let's forget about that here!). Here, for those of you who are not familiar with quantum mechanics, we will introduce the minimal concepts required to understand and practice (digital) quantum computing. Let's get started!

## 2.1 States

The state of a quantum mechanical system is its *most complete mathematical description.* The state is described by a vector in a Hilbert space.

Note: Throughout this course, states are pure states unless specified otherwise.

**Vectors:**
Dirac's notation is a popular notation in quantum mechanics used to denote vectors. A vector is denoted by a *ket* $|\psi\rangle$. What we write inside the ket $|.\rangle$ is just a symbol that we use to index or name the vector. The set of all such symbols is called the index set or alphabet of the vector space. The indices do not need to have precise mathematical meanings, but they can.

Examples:

- The ket $|1,0,0,1,1,0,0,1\rangle$ can describe the state of 8 qubits in a quantum computer. For now, we can think of qubits as simple two-state systems, with possible states 0 or 1.
- The state of a cat who does not possess the strongest chances of survival can be represented by the ket $|(\hat{}.\hat{})\rangle$ for alive and the ket $|({}^{x}.{}^{x})\rangle$ for dead.

**Vector spaces:**
A *vector space* is a set of vectors closed under scalar multiplication and addition operations. Formally, consider any two $d$-dimensional vectors $|\psi\rangle$ and $|\phi\rangle$ from a vector set $V \subseteq F^d$. Then, $V$ is a vector space if and only if all the linear combinations $\alpha |\psi\rangle + \beta |\phi\rangle \in V \ \forall \ \alpha, \beta \in F$ and $\forall \ \ |\psi\rangle, |\phi\rangle \in V$. In quantum mechanics, $F = \mathbb{C}$ and $d$ can also be infinite. However, in this course, we will deal with only finite values of $d$, mainly powers of 2, that is, $d = 2^n$ for integers $n \geq 1$.

Examples:

- The set of all position vectors $|x, y, z\rangle$, is the 3-dimensional Euclidean space $\mathbb{R}^3$. More generally, any set of vectors $|i\rangle \in \mathbb{C}^n$.
- In addition to vectors, matrices also form vector spaces. The set of all $m \times n$ matrices, $\mathcal{M}_{m\times n} \in \mathbb{C}^m \times \mathbb{C}^n$ forms a vector space.

For a given vector represented by a ket $|\psi\rangle$, we define its dual vector as a *bra* $\langle\psi|$.

**Hilbert space:**
A *Hilbert space* $\mathcal{H}$ is a special kind of vector space over the complex numbers $\mathbb{C}$. It is a vector space with an *inner product* (which is a generalization of the scalar product in the Euclidean space).

**Inner product**:
We define the *inner product* between two vectors $|\psi\rangle$ and $|\phi\rangle$ as a map from the Hilbert space $\mathcal{H}$ to the set of all complex numbers $\mathbb{C}$. That is, $\langle\psi|\phi\rangle \in \mathbb{C} \ \forall \ \ |\psi\rangle, |\phi\rangle \in \mathcal{H}$, with the following properties:

1. Positivity: $\langle\psi|\psi\rangle > 0$ for $|\psi\rangle \neq 0$
2. Linearity: $\langle\phi| (a |\psi_1\rangle + b |\psi_2\rangle) = a \langle\phi|\psi_1\rangle + b \langle\psi|\phi_2\rangle$
3. Skew symmetry: $\langle\phi|\psi\rangle = \langle\psi|\phi\rangle^*$

A Hilbert space is complete under the inner product induced norm $\| \, |\psi\rangle \, \| = \sqrt{\langle\psi|\psi\rangle}$. Completeness means that it contains all limiting vectors of absolutely convergent series. Complicated, but the good news is that this property is trivial for finite-dimensional Hilbert spaces.

*Remark:* Generally, quantum mechanics is defined in infinite dimensional Hilbert spaces. However, as we will see, finite-dimensional Hilbert spaces suffice to model and study digital quantum computing. We can therefore forget about the notion of completeness for the purpose of this course.

In a Hilbert space $\mathcal{H}$ of dimension $d$, we can define an orthonormal basis: $\{|\phi_1\rangle, |\phi_2\rangle, \ldots, |\phi_d\rangle\}$ such that $\langle\phi_j|\phi_k\rangle = \delta_{jk}$. The analogy in $\mathbb{R}^3$ are the three unit vectors $\hat{x}$, $\hat{y}$, and $\hat{z}$. Using this orthonormal basis, any vector $|\psi\rangle \in \mathcal{H}$ can be expanded in terms of the basis as:

$$|\psi\rangle = \sum_{j=1}^{d} c_j |\phi_j\rangle \, ; \quad c_j = \langle\phi_j|\psi\rangle \, . \tag{2.2}$$

It is important to notice that in quantum mechanics, the norm $\| \, |\psi\rangle \, \| = \sqrt{\langle\psi|\psi\rangle}$ of a state does not play any role in the theory (since we normalize states). Consequently, we can describe states as *rays*, namely vectors $|\psi\rangle$ with unit norm $\| \, |\psi\rangle \, \| = 1$. Also, a global complex phase is not physically relevant: $|\psi\rangle$ and $e^{\iota\alpha} |\psi\rangle$ are mathematically two different vectors but represent the same physical state. This will be clearer once we cover the measurement postulate.

Pay attention that this holds only for the overall phase factor $e^{\iota\alpha}$ on a vector. This means, $a\,|\phi\rangle + b\,|\psi\rangle$ and $a\,|\phi\rangle + e^{\iota\alpha} b\,|\psi\rangle$ are *not* the same physical state but $a\,|\phi\rangle + b\,|\psi\rangle$ and $e^{\iota\alpha}\left(a\,|\phi\rangle + b\,|\psi\rangle\right)$ are. Notice that if $|\phi\rangle = \sum_j a_j\,|\phi_j\rangle$ and $|\psi\rangle = \sum_j b_j\,|\phi_j\rangle$ are expanded in the orthonormal basis $|\phi_j\rangle$, then the inner product can be expressed in terms of the vector components $a_j$ and $b_j$ as $\langle\phi|\psi\rangle = \sum_{j=1}^{d} a_j^* b_j$.

Now, we shall describe operators, measurements and their properties governed by the postulates of quantum mechanics.

## 2.2 Measurements

An *observable* is a property of a physical system that can be measured. In quantum mechanics, observables are described by self-adjoint operators acting on the Hilbert space. A linear operator $\hat{A}$, is a linear map taking vectors to vectors: $|\psi\rangle \mapsto \hat{A}\,|\psi\rangle = |\phi\rangle$. Linearity implies $\hat{A}(a\,|\phi\rangle + b\,|\phi\rangle) = a\hat{A}\,|\phi\rangle + b\hat{A}\,|\phi\rangle$. Also, trivial but important, for two linear operators $\hat{A}$ and $\hat{B}$, the operator $\hat{C} = \hat{A}\hat{B}$ is the linear operator resulting from the application of $\hat{B}$ first, then $\hat{A}$. That is, $\hat{C}\,|\psi\rangle = \hat{A}\hat{B}\,|\psi\rangle \;\forall\; |\psi\rangle \in \mathcal{H} \iff \hat{C} = \hat{A}\hat{B}$.

The adjoint of an operator $\hat{A}$ is denoted by $\hat{A}^\dagger$ and is defined by its action as

$$\left\langle \phi \middle| \hat{A}\psi \right\rangle = \left\langle \hat{A}^\dagger\phi \middle| \psi \right\rangle \;\forall\; |\phi\rangle\,, |\psi\rangle \in \mathcal{H}. \tag{2.3}$$

Notice the ambiguous nature of dirac's notation here; $\hat{A}\,|\psi\rangle$ is denoted as $\left|\hat{A}\psi\right\rangle$.

A self-adjoint operator is such that $\hat{A}^\dagger = \hat{A}$. We will justify why self-adjoint operators are required to describe observables in quantum mechanics, but first lets consider their properties. If $\hat{A}$ and $\hat{B}$ are self-adjoint

1. $\hat{A} + \hat{B}$ is also self-adjoint.
2. $(\hat{A}\hat{B})^\dagger = \hat{B}^\dagger\hat{A}^\dagger = \hat{B}\hat{A}$ in general $\neq \hat{A}\hat{B}$.
3. $\hat{A}\hat{B} + \hat{B}\hat{A}$ and $i(\hat{A}\hat{B} - \hat{B}\hat{A})$ are self-adjoint.

Let's try to be a little bit less abstract. If we decompose a vector on a basis $|\psi\rangle = \sum_{j=1}^{d} a_j\,|\phi_j\rangle$, then we can always work with the mental image of the ket $|\psi\rangle$ as a column vector of its components.

$$|\psi\rangle \leftrightarrow \begin{pmatrix} a_1 \\ a_2 \\ . \\ . \\ . \\ a_d \end{pmatrix} \tag{2.4}$$

Hence, the dual vector or the bra $\langle\psi|$ is represented as the conjugate-transposed row vector.

$$\langle\psi| \leftrightarrow \begin{pmatrix} a_1^* & a_2^* & . & . & . & a_d^* \end{pmatrix} \tag{2.5}$$

A linear operator $\hat{A}$ acting on $|\psi\rangle$ can then be associated to a matrix.

$$\hat{A} \leftrightarrow \begin{pmatrix} A_{11} & A_{12} & A_{13} & ... & A_{1d} \\ A_{21} & A_{22} & A_{23} & ... & A_{2d} \\ . & . & . & & . \\ . & . & . & \ddots & . \\ . & . & . & & . \\ A_{d1} & A_{d2} & A_{d3} & ... & A_{dd} \end{pmatrix} \tag{2.6}$$

where the matrix elements $A_{jk}$ are given by

$$A_{jk} = \left\langle \phi_j \middle| \hat{A}\phi_k \right\rangle = \langle \phi_j | \hat{A} | \phi_k \rangle . \tag{2.7}$$

Consequently, a self-adjoint operator $\hat{A}$ is one for which the matrix is Hermitian, that is,

$$A_{jk} = A^*_{kj} \quad \forall \ k, j. \tag{2.8}$$

The application of an operator to a vector is then simply the matrix-vector multiplication

$$|\phi\rangle = A |\psi\rangle \tag{2.9a}$$

$$|\psi\rangle = \sum_{j=1}^{d} a_j |\phi_j\rangle \tag{2.9b}$$

$$|\phi\rangle = \sum_{j=1}^{d} b_j |\phi_j\rangle \tag{2.9c}$$

$$\hat{A} \leftrightarrow A_{jk} \tag{2.9d}$$

where,

$$\begin{pmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_d \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} & ... & A_{1d} \\ A_{21} & A_{22} & A_{23} & ... & A_{2d} \\ . & . & . & & . \\ . & . & . & \ddots & . \\ . & . & . & & . \\ A_{d1} & A_{d2} & A_{d3} & ... & A_{dd} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ . \\ . \\ . \\ a_d \end{pmatrix}, \tag{2.10}$$

concisely,

$$b_j = \sum_{k=1}^{d} A_{jk} a_k. \tag{2.11}$$

*Note:* The matrix-vector representation is by far the most intuitive representation as it clarifies almost all aspects. Further, it provides a tractable form to simulate quantum mechanics using computer programs like MATLAB. Hence, we strongly encourage and adapt this representation throughout this course.

Continuing, the inner product can be represented as a conjugate-row (the bra $\langle\phi|$) multiplying the column vector $|\psi\rangle$.

$$\langle \phi | \psi \rangle = (b^*_1, ..., b^*_d) \begin{pmatrix} a_1 \\ . \\ . \\ . \\ a_d \end{pmatrix} = \sum_j b^*_j a_j \tag{2.12}$$

Consequently, the norm can be written as

$$\|\psi\| = \sqrt{\langle\psi|\psi\rangle} = \sqrt{\sum_{j=1}^{d} |a_j|^2}. \tag{2.13}$$

Composite operators can be written as matrix-matrix multiplications

$$\hat{C} = \hat{A}\hat{B} \leftrightarrow \begin{pmatrix} C_{11} & C_{12} & ... & C_{1d} \\ C_{21} & C_{22} & ... & C_{2d} \\ . & \ddots & . & . \\ . & . & \ddots & . \\ C_{d1} & C_{d2} & ... & C_{dd} \end{pmatrix} = (A_{jk}).(B_{jk}) \tag{2.14}$$

$$C_{jk} = \sum_{l=1}^{d} A_{jl}B_{lk}. \tag{2.15}$$

Finally, we can build objects like a "ket-bra" represented by the outer-product or a column-row multiplication.

$$|\phi\rangle\,\langle\psi| \leftrightarrow \begin{pmatrix} b_1 \\ . \\ . \\ . \\ b_d \end{pmatrix} (a_1^*, ..., a_d^*) = \begin{pmatrix} b_1a_1^* & b_1a_2^* & ... & b_1a_d^* \\ b_2a_1^* & b_2a_2^* & ... & b_2a_d^* \\ . & \ddots & . & . \\ . & . & \ddots & . \\ b_da_1^* & b_da_2^* & ... & b_da_d^* \end{pmatrix}$$

**Eigenvectors and eigenvalues:**
For a square matrix $A \in \mathcal{M}_{dxd} : \mathbb{C}^d \to \mathbb{C}^d$, an eigenvector is a vector $|\nu\rangle \in \mathbb{C}^d$ such that the action of $A$ simply scales the vector: $A\,|\nu\rangle = \lambda\,|\nu\rangle$ where the eigenvalue $\lambda_j \in \mathbb{C}$.

Further, the following statements are equivalent.

1. $A$ is Hermitian: $A = A^\dagger$.
2. The eigenvectors of $A$ form an orthonormal basis that spans the entire space $\mathbb{C}^d$: $\text{span}\{|\nu_j\rangle\,|j = 1, 2, \ldots, d\}$ with $\langle\nu_j|\nu_k\rangle = \delta_{jk}$ where $A$ scales the eigenvctors $A\,|\nu_j\rangle = \lambda_j\,|\nu_j\rangle$.
3. The eigenvalues of $A$, are real: $\lambda_i \in \mathbb{R}\;\forall\; j = 1, 2, \ldots, d$.

Having introduced eigenvectors, eigenvalues and their properties for Hermitian matrices, we now mention an important theorem from linear algebra. This theorem will help us develop an intuition to aid us in digesting the measurement postulate a little better.

**Spectral theorem:**
In finite dimensions, any Hermitian matrix $A = A^\dagger \in \mathcal{M}_{dxd} : \mathbb{C}^d \to \mathbb{C}^d$ can be decomposed over its spectral basis or eigenbasis as

$$A = \sum_{i=1}^{d} \lambda_i |\nu_j\,\rangle\langle\,\nu_j|. \tag{2.16}$$

It can be a nice exercise to try and prove the above-stated properties and theorem.

Having covered all the prerequisites, we now describe the *act and consequence* of measurements in quantum mechanics.

1. **Outcome:** The outcome of measuring an observable $\hat{A}$ is always one of the eigenvalues $\lambda_j \in \text{spec}(\hat{A})$. The only possible outcome of a single act of measurement is an eigenvalue of $\hat{A}$. In quantum mechanics we deal with (a) observables that have a continuous spectrum, like position or momentum, (b) observables that have a discrete specturm, like angular momentum and (c) observables that have a mixed spectrum, partly discrete and partly continuous. However, for the purpose of this course and most of quantum computing, we deal with finite-dimensional observables that have discrete spectra.

2. **Probabilities:** The outcome or eigenvalue obtained through a measurement is determined by a (truly) random process, with a-priori probabilities. If the system is in state $|\psi\rangle$ and we measure $\hat{A}$ (whose eigenvalues are $\lambda_j$) then the probability of obtaining an outcome $\lambda_j$ is given by:

$$\text{prob}(\lambda_j) = \| \langle \nu_j | \psi \rangle \|^2 \tag{2.17}$$

   where $|\nu_j\rangle$ is the corresponding eigenvector. This is known as *Born's rule.* The above equation holds only in the case where $\lambda_j$ is a non-degenerate eigenvalue. That is, there exists no other eigenvector $\left|\nu_j'\right\rangle$ with the same eigenvalue $\lambda_j$. In the case of degenerate eigenvalues (not very important in quantum computing) Born's rule can be extended and the probability is defined more generally as:

$$\text{prob}(\lambda_j) = \|\hat{\Pi}_j |\psi\rangle \|^2 = \langle\psi| \hat{\Pi}_j |\psi\rangle \tag{2.18}$$

   where $\hat{\Pi}_j$ is the orthogonal projector on the sub-space spanned by the degenerate eigenvectors associated to $\lambda_j$. In the non-degenerate case, the orthonormal projectors are simply the rank-one projectors $\hat{\Pi}_j = |\nu_j\rangle\langle\nu_j|$. The orthonormality is defined as $\text{Tr}(\hat{\Pi}_j\hat{\Pi}_k) = \delta_{jk}$ and can be verified trivially for the rank-one projectors.

3. **Consequence:** Since we want measurements to be verifiable and hence repeatable, it is natural to require the state of the system after measurement to correspond to a state that justifies the outcome. In simpler words, measurements inevitably change the state of the system. The post-measurement state, when an outcome $\lambda_j$ is observed, has to be the corresponding eigenvector $|\nu_j\rangle$. Or, as is stated popularly, the system *collapses* onto the state $|\nu_j\rangle$. This is why the measurement postulate is also known as the *collapse postulate.* More generally, if $\lambda_j$ the outcome, then the new state is the projection onto the $\nu_j$ (maybe degenerate) eigenspace $\frac{\hat{\Pi}_j|\psi\rangle}{\|\hat{\Pi}_j|\psi\rangle\|}$.

Unlike the third point above, where we repeatedly measure on the same state, if we were to prepare several copies of the state $|\psi\rangle$ or to prepare a system in the state $|\psi\rangle$, perform a measurement $\hat{A}$ and then repeat the process several times, we would (potentially) get different outputs each time. This is owing to the inherent probabilistic (random) nature of measurements in quantum mechanics. Leveraging this, we can calculate the average or expectation value of the process.

$$\begin{aligned}
\langle \lambda \rangle &= \sum_j \lambda_j \ \text{prob}(\lambda_j) \\
&= \sum_j \lambda_j | \langle \nu_j | \psi \rangle |^2 \\
&= \sum_j \lambda_j \langle \psi | \nu_j \rangle \, \lambda_j \langle \nu_j | \psi \rangle \\
&= \langle \psi | \left( \sum_j \lambda_j \, | \nu_j \rangle \langle \nu_j | \right) | \psi \rangle \\
&= \langle \psi | \, \hat{A} \, | \psi \rangle
\end{aligned} \tag{2.19}$$

This is also known as the *expectation value of the operator* $\hat{A}$ on the state $|\psi\rangle$. We reiterate that this is not a characteristic that arises from the random errors due to an imperfect measurement apparatus, but is a fundamental property of quantum mechanics. Further, using discrete eigenvalues, we can completely rule out the role of imperfect measurements in generating this randomness. We simply need to use an apparatus with precision that is finer than the eigenvalue spacing.

Nonetheless, there exist some pairs or groups of observables that can be measured up to arbitrary precision individually but not simultaneously. These are known as non commuting observables–observables for which $\hat{A}\hat{B} \neq \hat{B}\hat{A} \implies [\hat{A}, \hat{B}] \neq 0$. The most famous example of such a pair is position and linear momentum, where $[\hat{x}, \hat{p}] = i\hbar$. However, such observables are of little importance right now and hence we shall not dwell over them.

Now, we go back to the question " *Why do we need self-adjoint operators for observables?*" We want an observable to be described such that, when we perform a measurement, we get a value that has some physical meaning. Since all (classical) measurements we make yield real values like position or momentum, it is preemptive but necessary to assume that observables obey the same in quantum mechanics and yield real eigenvalues. Further, whenever we perform a measurement using an observable, we want the observable to encompass all the possible outcomes and post-outcome consequences. That is, to have a complete eigenbasis that spans the entire Hilbert space. Hence, these two axiomatic requirements (motivated by classical mechanics and physical reality), lead to the representation of observables as Hermitian matrices. Although this section may seem very axiomatic and not concrete, quantum mechanics and especially the measurement postulate is by far the most accurate and widely tested description of nature.

Moving forward, we move to the rather not-so-unsettling topic of unitary time evolution in quantum mechanics, which unlike the measurement process, is determninistic. However, as we explain towards the end of the section, the non-unitary and non-deterministic nature of measurements poses a strange contradiction.

## 2.3 Time evolution

Every theory of physical phenomena must account for how systems evolve over time. In quantum mechanics, the state of the system changes over time through a *unitary operator*

$$|\psi(t)\rangle = \hat{U}(t, t_0)\,|\psi(t_0)\rangle \tag{2.20}$$

where $t_0$ is the initial time and $t \geq t_0$ is the final time. A unitary operator $\hat{U}$ has the property of preserving the norm and the inner product between states. Since $\hat{U}^\dagger \hat{U} = \mathbb{1}$,

$$\left\langle \hat{U}\phi \middle| \hat{U}\psi \right\rangle = \left(\hat{U}\,|\phi\rangle\right)^\dagger \left(\hat{U}\,|\psi\rangle\right) = \left(\langle\phi|\,\hat{U}^\dagger\right)\left(\hat{U}\,|\psi\rangle\right) = \langle\phi|\,\hat{U}^\dagger\hat{U}\,|\psi\rangle = \langle\phi|\,\mathbb{1}\,|\psi\rangle = \langle\phi|\psi\rangle \tag{2.21}$$

and

$$\|\hat{U}\,|\psi\rangle\,\| = \sqrt{\left\langle \hat{U}\psi \middle| \hat{U}\psi \right\rangle} = \sqrt{\langle\psi|\psi\rangle} = \|\,|\psi\rangle\,\|. \tag{2.22}$$

This latter property is a natural requirement since time evolution should not violate the physicality of a state and hence preserve its norm, since all physically allowed quantum states have unit norm.

Now, we look closely at how $\hat{U}(t, t_0)$ is defined. The *Schrödinger equation* governs time evolution in quantum mechanics as

$$\frac{d}{dt}\,|\psi(t)\rangle = -i\hat{H}(t)\,|\psi(t)\rangle\,, \tag{2.23}$$

where, we set the operator $\hat{H}$ in units of the reduced Planck's constant $\hbar = \frac{h}{2\pi}$. The operator $\hat{H}$ is called the Hamiltonian of the system and is the observable which can be used to measure the total energy. Solving Equation 2.23 for the case when the Hamiltonian $\hat{H}$ does not depend on time, we get

$$\hat{U}(t, t_0) = e^{-i(t-t_0)\hat{H}}. \tag{2.24}$$

The exponential of an operator is simply defined as the power series expansion of the operator involving powers of the operator. In the matrix representation, this would involve higher powers of matrices, which can be simply calculated by multiplying the matrix with itself several times.

As we will see later, in quantum computing, we will be interested in the specific time evolution of a few, namely one or two, quantum particles (qubits) under well-defined unitaries (gates) resulting from tailored Hamiltonians. Nonetheless, once we will define the universal set of these gates, we will not bother about their Hamiltonians or how they are realized in practice.

What we intend to highlight, is that unlike the measurement process that is inherently random, the dynamics or actions of all *quantum processes* are governed by deterministic unitary operators. This poses a contradiction and measurements need to be treated separately as special processes. This contradiction has been the topic of debate for almost a century and is a constant reminder of how we pay for the extraordinary predictive power of quantum mechanics by bearing with the awkwardness of the theory.

## 2.4 Composite systems

This is perhaps the most important postulate for quantum information, together with the measurement postulate. Given two systems $A$ and $B$, described in the Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$, the composite system, that is the total system describing both systems jointly belongs to the Hilbert space $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$. If system $A$ is prepared in a state $|\psi_A\rangle$ and system $B$ in a state $|\psi_B\rangle$, then the state of the composite system is:

$$|\psi\rangle = |\psi_A\rangle \otimes |\psi_B\rangle \tag{2.25}$$

where we use the definition of tensor products for vectors and matrices. If $\mathcal{H}_A$ and $\mathcal{H}_B$ have bases $\{|\phi_j\rangle_A ;\ j = 1, 2, \ldots, d\}$ and $\{|\phi_k\rangle_B ;\ k = 1, 2, \ldots, d\}$ respectively, then $\mathcal{H}$ is built by defining it's basis as $\{|\phi_j\rangle_A \otimes |\phi_f\rangle_B ;\ j, k = 1, 2, \ldots, d\}$.

States that can be written as $|\psi\rangle = |\psi_A\rangle \otimes |\psi_B\rangle$ are said to be "separable" as one can answer the question "*What is the state of subsystem A or subsystem B?*", separately for both $A$ and $B$. However, the linearity of quantum mechanics imopses that states like $|\psi\rangle = \alpha |\psi_A\rangle \otimes |\psi_B\rangle + \beta |\phi_A\rangle \otimes |\phi_B\rangle$ are also possible states of the composite system. These states are called non-separable or entangled states since here, we cannot write the state as a tensor product of two states and hence cannot find the states of the subsystems $A$ and $B$ separately.

Similar to tensor products of states, tensor products of operators are also possible. They are easily defined on separable states, and the general definition follows by linearity and expansion on a basis. For a separable operator $\hat{C} = \hat{A} \otimes \hat{B}$, given $|\psi_1\rangle = \hat{A} |\phi_1\rangle$ and $|\psi_2\rangle = \hat{B} |\phi_2\rangle$, then:

$$\begin{aligned} |\psi_1\rangle \otimes |\psi_2\rangle &= (\hat{A} |\phi_1\rangle) \otimes (\hat{B} |\phi_2\rangle) \\ &= \hat{A} \otimes \hat{B}(|\phi_1\rangle \otimes |\phi_2\rangle) \\ &= \hat{C}(|\phi_1\rangle \otimes |\phi_2\rangle) \end{aligned} \tag{2.26}$$

The tensor product is essential in quantum computing to describe a system made of several quantum bits.

## 2.5 The Quantum Bit

Let us get into business and define the elementary basis of quantum information. Analogous to the classical bit, which is a binary variable taking values 0 or 1, we define the quantum bit.

A qubit is the smallest non trivial quantum system, whose state is described by vectors in a Hilbert space of dimension two. The basis is called the *computational basis* and is composed of two orthogonal vectors $\{|0\rangle, |1\rangle\}$ with $\langle 0| |1\rangle = 0$ and $\| |0\rangle \| = \| |1\rangle \| = 1$. The simplest canonical representation is hence the 2D column vector basis

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.27}$$

The most general state of the qubit is then

$$|\psi\rangle = a |0\rangle + b |1\rangle \tag{2.28}$$

with the constraint $|a|^2 + |b|^2 = 1, a, b \in \mathbb{C}$.

### 2.5.1 Pauli Basis

The Pauli matrices play a central role in all of Quantum information. They are 2 × 2 complex matrices. They are both unitary and Hermitian, making them idempotent.

$$X = \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.29}$$

The eigenbases of each of these matrices are called the corresponding Pauli basis. It is simple to see that the canonical basis we chose previously is the Pauli $Z$ basis. These matrices have several interesting properties which we will see and use with due time (please see this Wikipedia page for now).

*Exercise:* We can try finding the Pauli $X$ and $Y$ bases.
*Solution:* They are given by

$$|\pm; X\rangle = \frac{|0\rangle \pm |1\rangle}{\sqrt{2}}, \quad |\pm; Y\rangle = \frac{|0\rangle \pm i\,|1\rangle}{\sqrt{2}}.e \tag{2.30}$$

But it is a healthy exercise to derive this yourself, at least once!

### 2.5.2 The Bloch Sphere

The Bloch sphere is a geometrical representation of a single qubit's state space. That is, all single qubit states lie on or inside the Bloch sphere. In this and the following few chapters, it will be sufficient for us to address only the surface of the sphere (pure states), and then we shall study mixed states later. Now, we show exactly how pure states are represented on the Bloch sphere. We begin with rewriting the general single qubit state as

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle\,. \tag{2.31}$$

This representation is completely general and equivalent to the one presented in Equation 2.28. For $\theta \in [0, \pi], \phi \in [0, 2\pi]$, we can span all possible values of the coefficients $a, b \in \mathbb{C}$. The geometric nature arises naturally from the fact that we can now represent $\theta$ and $\phi$ as angles made by a vector on a sphere. Specifically,

- We choose the 3D Euclidean space.
- Due to the normalization condition, all states are represented by unit vectors residing on the surface of the sphere.
- The angles $\theta$ and $\phi$ are the angles made by the vector (state representation) with the $+z$ and $+x$ axes respectively. This combined with Equation 2.31 makes it easier to see why it is sufficient to choose the range of $\theta$ only up to $\pi$.
- Hence, the Pauli $Z$ basis states $|0\rangle, |1\rangle$ are represented as unit vectors along the $+z$ and $-z$ axes respectively with $\theta_{|0\rangle} = 0$ and $\theta_{|1\rangle} = \pi$. Similarly, the Pauli $X$ and $Y$ basis states are represented along the $\pm x$ and $\pm y$ axes, respectively.

This representation will be particularly useful later for understanding crucial theorems on single qubit transformations.

*Exercise:* We can develop familiarity with the Bloch sphere representation by computing the angles $\theta$ and $\phi$ for each of the Pauli basis states. Visualizing them on the sphere

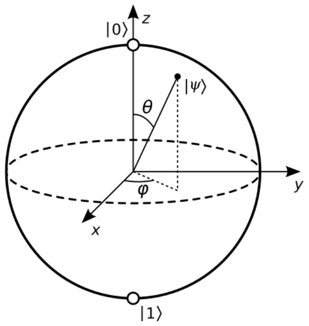


Figure 2.1: The Bloch sphere representation.[1]

might help us develop a better geometrical intuition.
Solution: For the $X$ basis, $\theta_{|\pm,X\rangle} = \frac{\pi}{2}$, $\phi_{|+,X\rangle} = 0$, $\phi_{|-,X\rangle} = \pi$ and for the $Y$ basis $\theta_{|\pm,Y\rangle} = \frac{\pi}{2}$, $\phi_{|+,Y\rangle} = \frac{\pi}{2}$, $\phi_{|-,Y\rangle} = \frac{3\pi}{2}$.

### 2.5.3 A brief discourse on measurement

In quantum computing, we will assume that measurements are carried out in the computational basis $\{|0\rangle, |1\rangle\}$ unless specified otherwise. Hence, when measuring on a state $|\psi\rangle = a|0\rangle + b|1\rangle$ one obtains "0" or "1" with probability $\| \langle 0|\psi\rangle \|^2 = |a|^2$ and $\| \langle 0|\psi\rangle \|^2 = |b|^2$, respectively. The state after measurement will be $|0\rangle$ or $|1\rangle$ depending on the outcome of the measurement.

Notice that in a single *shot* (act of measurement), it is impossible to know the values of $a$ and $b$, and thus the state. After the 1$^{\text{st}}$ measurement, the state is lost. This is the impossibility of completely knowing a quantum state from just one instance of it. This is related to the no-cloning theorem, which prohibits us from developing one single apparatus that can clone any arbitrary quantum state. If we could clone $|\psi\rangle$, without knowing it fully, then we could generate several copies, measure them separately and learn more information about $|\psi\rangle$.

Finally, if in one measurement, we cannot obtain anything except $|0\rangle$ or $|1\rangle$, it would be natural to question if a qubit is any different from a classical probabilistic binary bit? Yes, it is because of entanglement, which we will see in more detail soon.

### 2.5.4 Multiple Qubits

A quantum computer consists of several qubits. Thanks to Postulate IV, we know how to describe the entire system. If $\mathcal{H}_1$ is the Hilbert space of one qubit, then for $n$ qubits, the total composite Hilbert space is given by

$$\mathcal{H}_n = \overbrace{\mathcal{H}_1 \otimes \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_1}^{n \text{ times}} = \mathcal{H}_1^{\otimes n}. \tag{2.32}$$

$\mathcal{H}_n$ is the Hilbert space corresponding to the entire quantum system on which quantum computations will be performed. This is a space of dimension $d = 2^n$ spanned by the

[1] The state vector $|\psi\rangle$, lies on the surface of the sphere. However, since we projected a 3D sphere onto this 2D page, it appears as if the vector is shorter.

composite (tensor product) computational basis of length $2^n$.

$$\begin{aligned}\{|0\rangle \otimes |0\rangle \otimes ... \otimes |0\rangle\,,\ |0\rangle \otimes |0\rangle \otimes ... \otimes |1\rangle\ \ |1\rangle \otimes |1\rangle \otimes ... \otimes |1\rangle\} = \\ \{|00...0\rangle\,,\ |00...01\rangle\,,\ |00...10\rangle\,, ...,\ |11...1\rangle\},\end{aligned} \tag{2.33}$$

where the last shortcut notation is adapted without the loss of generality. We notice that the basis indices now look like binary bit-strings of length $n$. Leveraging this, we succinctly represent the basis as $\{|j\rangle\,;\ j = 0, 1, 2, \ldots, 2^n - 1\}$ by replacing the binary bit-strings inside the kets with the corresponding binary integer. Explicitly, $j = x_0 \times 2^0 + x_1 \times 2^1 + \cdots + x_{n-1} \times 2^{n-1}$. Finally, a general multi-qubit state is a linear superposition of these basis elements.

$$|\psi\rangle = \sum_{j=0}^{2^n-1} c_j\, |j\rangle \tag{2.34}$$

The probability of getting the state $|j\rangle$ as the measurement outcome upon measuring in the $n$-qubit computational basis is $\|\, \langle j|\psi\rangle\, \|^2 = |c_j|^2$.

The exponentially large size of the multi-qubit Hilbert space should give us an idea why we cannot simulate arbitrary quantum systems on classical computers efficiently. More importantly, the exponentially large span of $n$ qubits as opposed to $n$ classical bits, gives us a sliver of hope that we can use quantum computers to also solve classically intractable problems with (hopefully) exponentially smaller system sizes. However, there is no free lunch and hence, if we want to extract complete information about an $n$-qubit state then we need to perform exponentially many measurements.

The art of quantum computation lies in designing algorithms that leverage quantum effects like interference and entanglement through smart circuitry or measurement schemes to avoid exponential overheads while successfully encoding and extracting the information required to solve the problem. In the following lectures, we will see some great examples of this philosophy in action.

In this course, we ignore the practical aspects involve in physically realizing such systems. Physically realizing such a system would require one to develop $n$ qubits and then control all the interactions between them. This, in practice, is a challenging task, which becomes more arduous upon pairing with the perils of errors and noise, as we will see later.

# Chapter 3

# The paradigm of digital Quantum Computation

The goal of digital quantum computation is to devise a universal paradigm for computation leveraging the application of unitary operations on a set of qubits. Similar to classical computation, we would like this to be feasible in practice, which requires defining the elementary building blocks. Further, we would like these building blocks to be simple to engineer, control and measure/read out. This decomposition is essential for both mathematical and physical reasons. Mathematically, it makes it possible to clearly establish the notion of quantum computational complexity. Physically, it fulfills the locality criterion: we build complex gates using elementary gates that are local in space as they correspond to physical objects with local interactions.

| **Classical Computation** | **Quantum Computation** |
|---|---|
| Processes bits. | Processes qubits. |
| Made from a finite set of logical gates (universal gates). | Made of a finite set of quantum logical gates (universal gates). |
| Relies on the initialization of the input to an arbitrary value. | Not all initial states are easy to prepare. |
| Allows the possibility of reading out the output. | Allows readout through quantum measurements and collapse. |
| Relies on error correction (or on few errors). | Quantum error correction (no-cloning theorem). |

Table 3.1: Comparison between classical and quantum computation.

Since quantum computing is a superset of classical computing, the quantum computation paradigm must fulfill several requirements. Foremost, it must contain classical computation, and it must reproduce it efficiently, namely at a maximal polynomial overhead (in time and space). This is true because:

1. **Statement 1**: Irreversible quantum computing (AND, XOR, etc.) can be reproduced by reversible classical computing.
   *Proof*: $f: \{0,1\}^n \to \{0,1\}^m$ with $m < n$: irreversible.
   *Define*: $\tilde{f}: \{0,1\}^{n+m} \to \{0,1\}^{n+m}$, $\tilde{f}(x; 0^{(m)}) = (x; f(x))$.

2. **Statement 2**: It can be done efficiently. For proof, see [1].
   Statements 1 and 2 ensure that the notion of classical computational complexity is not dependent on the paradigm, whether reversible or irreversible, but it's universal.

3. **Statement 3**: Classical reversible computing is a particular case of quantum computing.
   *Proof*: Recall that quantum computing is a unitary operation $U : \mathcal{H}^{\otimes n} \mapsto \mathcal{H}^{\otimes n}$. A reversible classical operation on $n$ bits sends each bitstring $x$ into a bitstring $y$ and consequently, the total result of $f(x)$ is a permutation of the set of $2^n$ input bitstrings. It suffices to define $U : |x\rangle \mapsto |y\rangle$ as a permutation of the vectors of the computational basis. Essentially, $U$ is a $2^n \times 2^n$ matrix that has all zeros and one 1 in each row and each column. The question remains open if we can execute an arbitrary unitary $U$ with the quantum computing paradigm efficiently (we will answer this soon).

*Reminder*: The state of 1 qubit is defined in $\mathcal{H}_1$ with computational basis $\{|0\rangle, |1\rangle\}$. $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|\alpha|^2 + |\beta|^2 = 1$. Consequently, the state of $n$ qubits is defined in $\mathcal{H}_n = \mathcal{H}_1 \otimes \mathcal{H}_1 \otimes ... \otimes \mathcal{H}_1 = \mathcal{H}_1^{\otimes n}$ with computational basis $\{|00...0\rangle, |00...01\rangle, ..., |11...1\rangle\}$.

## 3.1 Quantum gates

Quantum gates are the general unitary operations applied to any quantum circuit. These gates can act on single or multiple qubits at the same time. For an arbitrary state $|\psi\rangle = \sum_j c_j |j\rangle$, where $\{|j\rangle\}$ is the computational basis, the action of any unitary $U$ on the state can be written as

$$U|\psi\rangle = U\left(\sum_j c_j |j\rangle\right) = \sum_j c_j U(|j\rangle) \tag{3.1}$$

where, the second step leverages the fact that $U$ is a linear operator. Hence, to completely define a gate, it is sufficient to study its actions on the basis elements. Then its action on any other state can be determined by expanding the state as a linear combination over the basis states and the output state is simply the same linear combination of the outputs corresponding to each basis state. This brings us to the topic of single and multiple qubit gates.

### 3.1.1 Single-qubit gates

Single-qubit gates are simple unitary operations $U$ acting on the Hilbert space of one qubit $\mathcal{H}_1$. The elementary single qubit quantum gates are

1. The Pauli gates.

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

We recall their commutator relations

$$\begin{aligned}[X,Y] = XY - YX &= 2iZ \\ [Y,Z] &= 2iX. \\ [Z,X] &= 2iY\end{aligned}$$

2. Hadamard gate: $H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$
3. Phase gate: $S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$
4. $\pi/8$ or T-gate: $T = \begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}$

Now, it is natural to ask why these gates are important. Pauli gates play a special role in physics. They are used to generate rotation operators (and also in error corrections):

$$R_\alpha(\theta) = e^{-i\theta\alpha}\ \forall\ \alpha = X, Y, Z. \tag{3.2}$$

Rotations are useful in proving the theorem on realizing arbitrary unitaries. They are also important in defining some important algorithms, such as the QFT (Quantum Fourier transform). We will see both these concepts soon.

Notice that $S = T^2$. So, why do we have to include both in our set of universal gates? The reason is rather deep and concerns quantum error correction. Hence, it will be clearer later in the course. For now, it is better that we keep both and use $S$ instead of $T^2$ whenever possible. Next, we shall look at some useful theorems.

### 3.1.2 Some useful theorems

First, we will understand how arbitrary single-qubit gates act. Any single qubit gate that maps any pure quantum state to another essentially performs a rotation on the Bloch sphere (up to an arbitrary phase $e^{i\alpha}$). This idea helps us see two natural representations of an arbitrary single-qubit gate.

**Theorem 1:** An arbitrary single qubit gate $U$ can be written as a rotation by an angle $\theta$ about some direction given by a unit vector $\hat{n}$.

$$\begin{aligned}U &= e^{i\alpha}R_{\hat{n}}(\theta) \\ &= e^{i\alpha}e^{-i\theta(n_xX + n_yY + n_zZ)}\end{aligned} \tag{3.3}$$

**Theorem 2:** An arbitrary single qubit gate $U$ can be written as the composition of three independent rotations along the $x$, $y$ and $z$ directions by angles $\beta, \gamma, \delta \in [0, 2\pi]$, respectively.

$$U = e^{i\alpha}R_z(\beta)R_y(\gamma)R_z(\delta) \tag{3.4}$$

Both these representations are natural to see if we visualize states and operations on the Bloch sphere. The unitaries $R_x, R_y$ and $R_z$, represent rotations around the $x, y$ and $z$ axes respectively. This makes it obvious that any rotation around the Bloch sphere can be now written as a composition of rotations around the three axes, or as a single rotation around an intermediate axis given by $\hat{n}$.

**Corollary:** An arbitrary single-qubit gate U can be written as:

$$U = e^{i\alpha} AXBXC \tag{3.5}$$

where $A, B$ and $C$ are unitary matrices and $ABC = \mathbb{1}$ *Proof*: It is fairly simple to see this by setting

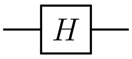
$$\begin{aligned} A &= R_z(\beta)R_y(\gamma/2) \\ B &= R_y(-\gamma/2)R_z(-(\delta+\beta)/2) \\ C &= R_z((\delta-\beta)/2) \end{aligned} \tag{3.6}$$

in Theorem 2. This mysterious relation will prove its usefulness in a moment.

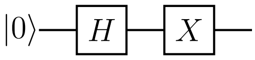
The two theorems above can be tied to the fact that the Pauli matrices are generators of the Lie group $SU(2)$, which is the group of all possible single qubit unitaries. For brevity of these notes, we skip the details about generators and groups here, but would be happy to provide the sources if it interests you :)

### 3.1.3 Quantum circuit notation

Each qubit is represented by a line or wire. Most of the time, gates are represented by boxes. Consider a simple circuit below that applies the Hadamard gate on a single qubit.



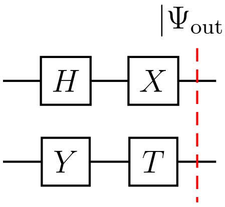
Successive quantum operations are denoted by successive gates. However, we need to be careful that gates are applied from left to right (first to last), whereas the operators are applied from right to left (first to last) when acting on a quantum state. For example, the circuit for implementing the Hadamard gate followed by the Pauli-$X$ gate on a single qubit is



and corresponds to the operation $XH|0\rangle$.

*Note:* It is customary to consider the initial state of each qubit in the circuit to be $|0\rangle$ unless specified otherwisde.

Now consider a two-qubit circuit with single-qubit gates.



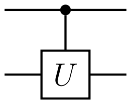
The output is a product state given by the tensor product between the output states of the two qubits $|\Psi_{\text{out}}\rangle = XH|0\rangle \otimes TY|0\rangle$.

### 3.1.4 Two-qubit gates

Extending single-qubit gates, we can also define two, there or multiple qubit gates. However, most multiqubit ($\geq 3$) gates can be constructed using two-qubit gates. The most interesting two-qubit gates are Controlled gates C-$U$. These gates are controlled by a control qubit; that is, the gate $U$ is applied to the target qubits if the control qubit is in state $|1\rangle$. Hence, such a gate can be expressed as

$$\text{C-}U = |0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes U. \tag{3.7}$$

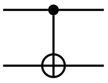
And in the circuit representation they are denoted as



where the black dot on the first qubit indicates that it is the control qubit.

Controlled gates are immensely useful in almost all multi-qubit operations and play a pivotal role in generating entanglement in quantum circuits (as we will see soon). We are mostly interested in two controlled operations, C-NOT and C-$Z$, which, as the names suggest, apply a controlled-NOT ($X$) gate and a controlled-$Z$ gate, respectively.

**C-NOT:** It is a 2-qubit operation defined on the computational basis in a way that it flips the 2nd qubit if the first qubit is in the state $|1\rangle$ and does nothing if the 1st qubit is in the state $|0\rangle$. It is represented in a circuit as follows.



Further, its matrix representation is given by studying its action on the computational basis through its truth table as shown in Table 3.2.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Concisely, it can be expressed in the computational basis as

$$|c\rangle\,|t\rangle \rightarrow |c\rangle\,|t \oplus c\rangle\,, \tag{3.8}$$

where "$c$" and "$t$" denote the states of the control and target qubits, respectively. The $\oplus$ symbol denotes the addition modulo 2: $0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$ and $1 + 1 =$

| Input state | Output state |
|---|---|
| $\|00\rangle$ | $\|00\rangle$ |
| $\|01\rangle$ | $\|01\rangle$ |
| $\|10\rangle$ | $\|11\rangle$ |
| $\|11\rangle$ | $\|10\rangle$ |

Table 3.2: Truth table for the C-NOT gate.

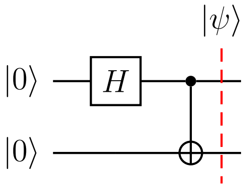
$(2\text{modulo}2) = 0$, which is denoted formally as $2 \equiv 0(\text{mod}2)$. This also motivates why the circuit representation is that of a controlled $\oplus$ operation.

A particularly useful representation of the C-NOT gate is

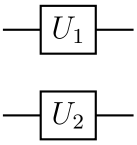
$$\text{C-NOT} = |0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes X. \tag{3.9}$$

This representation clearly indicates that it is the sum of two tensored operators, and hence its corresponding unitary $U_{\text{C-NOT}}$ is a non-separable 2-qubit unitary. That is, it can not be written as $U_{\text{C-NOT}} = U_1 \otimes U_2$ where $U_{1,2}$ are single qubit unitaries that act on qubits 1 and 2 respectively. The fact that it is non-separable is important as it *can generate entanglement.*

*Example:*



where, $|\psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. On the other hand, a separable circuit of two qubits can not.

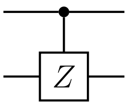


This is because the output state in the latter case can be simply written as $U_1|0\rangle \otimes U_2|0\rangle$. Hence, single-qubit gates are said to be local and can be implemented relatively easily on physical systems as compared to multi-qubit non-local operations.

Due to their non-local nature, two-qubit operations pose a challenge while physically realizing quantum computers. Since entanglement requires physical interaction between the entangled entities, a two-qubit non-separable operation requires physical interaction between qubits. Naturally, local interactions between neighboring qubits are stronger and more persistent than long-range interactions. Hence, to entangle any arbitrary pairs of qubits, the topology of the set of qubits on a quantum processor plays a pivotal role. For example, Google's Sycamore quantum processor is a square grid of qubits in which each qubit can interact (and thus implement gates like C-NOT) with its four neighbors only. Now, we look at another interesting gate, the controlled phase gate or the C-$Z$ gate.

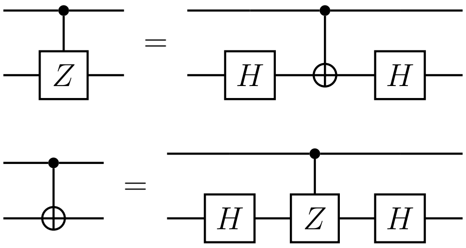
**C-$Z$:** Similar to the C-NOT gate, the C-$Z$ gate applies the Pauli $Z$ operator on the target qubit if the control qubit is in state $|1\rangle$ or leaves the target qubit unchanged if the control qubit is in state $|0\rangle$.

$$\text{C-}Z =$$

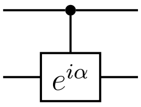


$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} = |0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes Z.$$

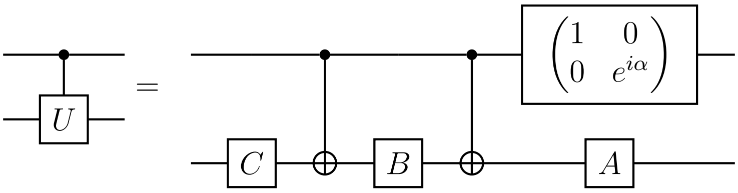
Since the Pauli $X$ and $Z$ operators are related by the basis transform (using the Hadamard unitary operation), $X = HZH$ and $Z = HXH$, we can write the controlled operations C-NOT and C-$Z$ in terms of each other as follows.



Some controlled operations can also be separable. For example,

C-$\alpha$ =  $= |0\rangle\langle 0|\otimes\mathbb{1}+|1\rangle\langle 1|\otimes e^{i\alpha}\mathbb{1} = \begin{pmatrix}\mathbb{1} & 0\\ 0 & e^{i\alpha}\mathbb{1}\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & e^{i\alpha}\end{pmatrix}\otimes\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} = U(e^{i\alpha})\otimes\mathbb{1}.$

What about a general controlled unitary operation C-$U$? These play a very important role in all the quantum algorithms. Hence, it is imperative to address the question, if for an arbitrary single-qubit unitary operation $U$), can we easily build a circuit for applying the controlled operation C-$U$? Recalling the result $U = e^{i\alpha}AXBXC$ with $ABC = \mathbb{1}$, we can implement the controlled operation as follows.

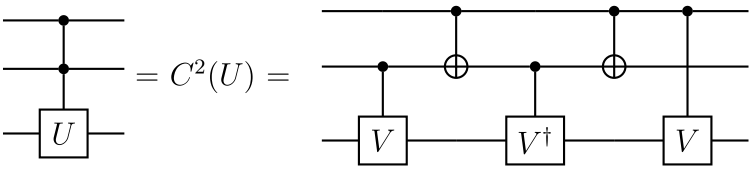


*Exercise:* We leave it as an exercise to verify this result.
*Hints:* (a) Try using the definition of C-$U$ as a sum of tensor products and perform the full matrix multiplication or (b) simply check if the action of the given circuit matches the desired action of C-$U$ on all the four 2-qubit basis states.

### 3.1.5 Universal gates

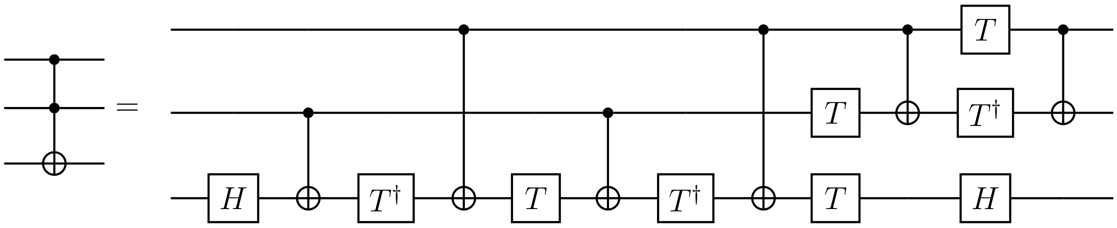
Universal gates are a set of gates that can be used to construct any other gate. The Hadamard, Phase and $T$ gates, along with the two-qubit C-NOT gate, form a universal set of quantum gates. We will see this in more detail in this section.

Using C-NOT and arbitrary single-qubit operations, we can make any arbitrary C-$U$. Further, we can also build doubly-controlled unitary gates using C-$U$ and C-NOT gates



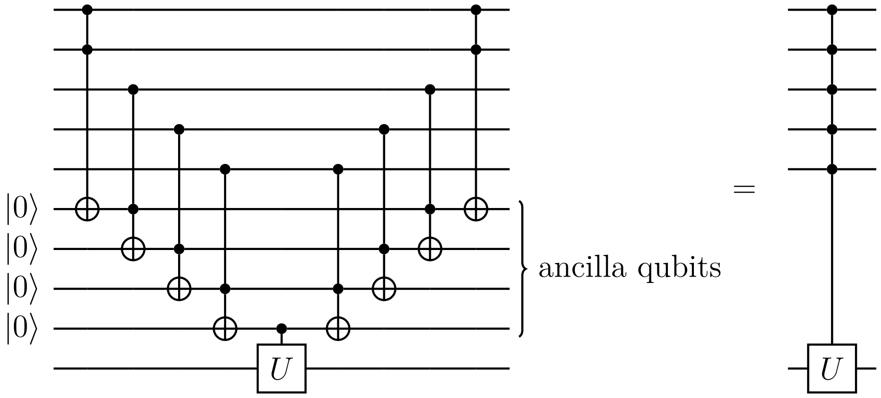
for any V such that $V^2 = U$.

The special gate $C^2(X)$ (doubly-controlled NOT) can be realized using $V = (1-i)(\mathbb{1} + iX)/2$. The explicit circuit for $C^2(X)$ is given below.



This gate is very important and is called a Toffoli gate. The Toffoli gate plays a fundamental role in establishing a relation between classical and quantum computation. It is shown that universal reversible classical computation requires a three-bit gate since one and two-bit gates are not enough. This three-bit gate can be chosen to be the Toffoli gate. However, when we go from classical to quantum computing and consider the very same gate, it is no longer fundamental and can be expressed in terms of 1 and 2-qubit gates instead. This fact provides a glimpse of the advantage of quantum computing over classical computing.

Iteratively continuing the above line of thought, the construction of multi-controlled quantum gates can be realized. For example, $n$-qubit controlled operations $C^n(U)$ gates can be decomposed as ladder-like arrangements of C-NOT gates.



This example is instructive for two reasons.

1. It exemplifies the notion and importance of *ancilla qubits*, which are an additional set of qubits we can use as a help for realizing the target operation. Often, we would like to avoid these ancilla qubits, but sometimes, they are unavoidable.

2. It demonstrates the idea of *uncomputation*. To achieve the operation $C^n(U)$, we make operations on the ancilla qubits too (the NOTs from the $C^2(X)$ gates). We may leave the ancilla qubits as is after the C-$U$. However, we usually undo the operations on the ancilla qubits because we may have to use them again for further calculations, and it is often more efficient to "reuse" qubits than to use a larger set of qubits.

We may now ask "*How efficiently can this paradigm help us implement an arbitrary unitary operation $U$ on $n$ qubits?*". What we know is:

1. A singly-controlled C-$U$ can be built efficiently using C-NOT and $V = \sqrt{U}$ gates.

2. A multi-controlled $C^n(U)$ can be built efficiently from C-NOT and C-$U$) gates.

Hence, we can prove the following statement (please refer [2] for more details).

**Statement 1:** Any arbitrary unitary $U_n$ acting on $n$ qubits can be implemented exactly using $\mathcal{O}(n^2 4^n)$ arbitrary single-qubit gates and C-NOT gates.

*Proof:* The proof of this statement is given in [2] (Section 4.5.2). Further, it is shown that this construction is optimal in the worst-case scenario, that is, for the hardest $n$-qubit unitaries, we cannot do better than $\mathcal{O}(n^2 4^n)$ C-NOTs and single qubit unitaries.

The resources required to implement an arbitrary $n$-qubit unitary $U_n$ grow exponentially with the number of qubits $n$. This very clearly establishes that quantum computing is not universally efficient; that is, we cannot expect to solve arbitrary problems with sub-exponential overheads. Consequently, this necessitates the need to design algorithms that leverage quantum effects without introducing exponential overheads.

Up until now, we assumed access to arbitrary single-qubit unitaries $U$ and hence using C-NOT (for multi-qubit extension) was enough to implement arbitrary multi-qubit unitaries[1]. However, we now highlight a key subtlety overlooked when discussing the implementation of arbitrary single-qubit unitaries. Although we showed earlier that any single qubit unitary can be decomposed using rotations around the three cartesian axes, we glossed over the fact that exactly implementing arbitrary rotations around the axes is not trivial. In fact, this too, is exponentially hard to do when equipped with a finite gate set. So are we doomed? Of course not. If quantum computing was truly more expensive than useful, we would not be studying this subject today.

If not exactly, then "*Can we approximate an arbitrary single-qubit unitary $U$ with an efficiently long sequence of gates taken from a finite set?*" The short answer is yes! Formally, using the distance between two unitary operators, we define the problem as follows.

$$D(U,V) = \max_{|\psi\rangle} \|(U-V)\,|\psi\rangle\,\|, \tag{3.10}$$

where, $|\psi\rangle$ are normalized states in the Hilbert space of $n$ qubits. This definition of distance is physically sound as; indeed, it implies that if we measure an observable $\hat{O}$ on $U\,|\psi\rangle$ and $V\,|\psi\rangle$, then the probabilities $p_U$ and $p_V$ to obtain a given outcome have the relation

$$|p_U - p_V| \leq 2D(U,V). \tag{3.11}$$

Hence, if we want to approximate $U$ with a sequence of gates from the universal set $S$ with accuracy $\epsilon$, that is, for $V = V_1 V_2 \dots \ \forall\ V_i \in S$, it would imply that $E(U,V) < \epsilon$.

Let's start by establishing a minimal *universal gate set.* That is, a set of gates that can be used to implement any required operation (up to arbitrary precision). In classical computing, these sets are single gates, either NAND or NOR. In quantum computing, the most famous universal gate set is the *Clifford+$T$* gate set consisting of the $H$, $S$, C-NOT and $T$ gates.

The Solovay-Kitaev Theorem states that any arbitrary 1-qubit gate can be approximated with accuracy $\epsilon$ using a sequence of $\mathcal{O}([\log(1/2)]^\tau)$ gates from the single-qubit universal gate set $\{H, S, T\}$, with $\tau \sim 2$. Consequently, A circuit containing $m$ C-NOT gates and single qubit unitaries can be implemented with accuracy $\epsilon$ using $\mathcal{O}(m[\log(m/\epsilon)]^\tau)$ gates from Clifford+$T$ gate set. This scaling is efficient as it scales poly-logarithmically in $1/\epsilon$.

[1] A group at ETHZ has developed a technique to directly implement C-$U_1$ operations without decomposing them into C-NOTs and $U$s.

Unfortunately, the other step, namely going from arbitrary $n$-qubit $U_n$ to 1-qubit unitaries and C-NOTs is not efficient as mentioned before.

> *The digital quantum computation paradigm is not universally efficient and requires clever algorithmic design choices to achieve satisfactorily accurate soltions.*

## 3.2 Quantum state preparation

Another requirement of the digital paradigm of quantum computation is to prepare the input to a quantum algorithm. If a quantum circuit has $n$ qubits, then the most general input state is an arbitrary quantum state of $n$ qubits:

$$|\psi_{\text{in}}\rangle = \sum_{n=0}^{2^n-1} \alpha_n |n\rangle ; \quad \sum_{n=0}^{2^n-1} |\alpha_n|^2 = 1 \tag{3.12}$$

Similar to implementing an arbitrary unitary, preparing an arbitrary quantum state is not a task that can be carried out efficiently, even on a quantum computer. Here, we only provide intuitive arguments and recommend references like [3] for rigorous statements.

One obvious way to prepare an arbitrary input state $|\psi_{\text{in}}\rangle$ would be to start from a standard state, say $|0\rangle$, and use a unitary transformation:

$$|\psi_{\text{in}}\rangle = U_{\text{in}} |0\rangle \tag{3.13}$$

where the state preparation unitary $U_{\text{in}}$, can be decomposed into elementary operations of the quantum computer itself. However, as we have seen, this decomposition is not efficient in the most general case. It can be shown that an arbitrary state preparation unitary $U_{in}$, requires $O(2^n)$ quantum gates to approximate it [3].

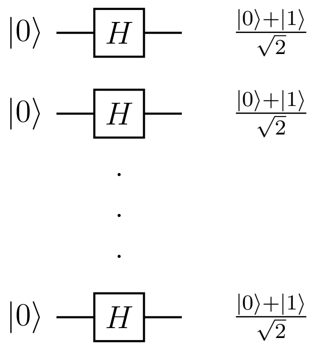
An important remark is that the complexity of preparing an arbitrary state need not be the same as that of preparing an arbitrary unitary. For state preparation, we have the additional freedom that we can choose the unitary $U_{in}$ and the standard state $|\psi_0\rangle$ we start from. Hence, it might be less complex to prepare a state than to implement an arbitrary unitary. nevertheless, preparing an arbitrary state $|\psi\rangle$ may require $O(2^n)$ operations in most cases. Both statements are proven by a simple counting argument about the number of final combinations and the finiteness of the set of quantum gates.

Therefore, for quantum computing, we will simply assume to always start from $|00...0\rangle$. Again, the skill of quantum scientist is that of devising useful algorithms that require as an input a state $|\psi_{\text{in}}\rangle$ that can be prepared efficiently from $|00...0\rangle$, using additional (but efficient) initialization circuits.

One example that occurs frequently is the total superposition state:

$$|\psi_{\text{in}}\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle . \tag{3.14}$$

This state can be efficiently prepared using only $n$ Hadamard gates.


$|0\rangle$
H
$\frac{|0\rangle+|1\rangle}{\sqrt{2}}$
$|0\rangle$
H
$\frac{|0\rangle+|1\rangle}{\sqrt{2}}$
$|0\rangle$
H
$\frac{|0\rangle+|1\rangle}{\sqrt{2}}$

$$|\psi_{\text{in}}\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes n} = \frac{1}{\sqrt{2^n}} \left(|00...0\rangle + |00...1\rangle + |11...1\rangle\right) \tag{3.15}$$

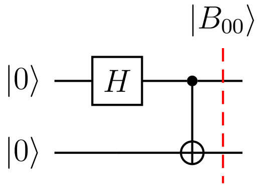
This state is useful in achieving *quantum parallelism* (more on this in the next lecture) because any function $f : \{0,1\}^n \to \{0,1\}^n$, when applied to $|\psi_{\text{in}}\rangle$, is applied *simultaneously* (in superposition) to all possible classical inputs $|n\rangle$.

### 3.2.1 Bell states

Finally, it is important to note that the complexity of preparing a state is not related to its level of entanglement. There are examples of highly entangled states that admit quite simple preparation circuits, like the Bell states.

$$\begin{aligned} \left|\Phi^+\right\rangle = |B_{00}\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right), \quad \left|\Phi^-\right\rangle = |B_{10}\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) \\ \left|\Psi^+\right\rangle = |B_{01}\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right), \quad \left|\Psi^-\right\rangle = |B_{11}\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right) \end{aligned} \tag{3.16}$$

The indexing used is quite handy since it allows us to transform among the Bell states using $|B_{ij}\rangle = Z^i_{(0)} X^j_{(1)} |B_{00}\rangle$, where the subscripts represent the qubit on which the gate acts. The state $|B_{00}\rangle$ can be easily prepared by applying a C-NOT after a Hadamard gate on the state $|00\rangle$.



## 3.3 Readout

The act of measuring the outputs of a quantum circuit is known as readout. Reading out the results from a quantum circuit must be done through quantum measurement. Owing to the measurement postulate, (most of the time) this will result in state collapse and thus loss of quantum information initially stored in the output state $|\psi_{\text{out}}\rangle$.

One strategy to improve this scenario is to devise probabilistic quantum algorithms–algorithms that are executed repeatedly from the same initial state $|\psi_{\text{in}}\rangle$, and the result of the calculation is inferred statistically by measuring the output after each run. As we will see, most quantum algorithms are probabilistic.

In general, we might want the ability to measure an arbitrary observable $\hat{O}$ on the output state $|\psi_{\text{out}}\rangle$. However, as we will see, similar to the input state preparation, we don't necessarily have to measure $\hat{O}$.

We see this by showing how we can compute the expectation value or the probabilities of the different outcomes while measuring an observable $\hat{O}$ on a state $|\psi\rangle$ using a quantum computer. We know that a measurement of $\hat{O}$ will give an eigenvalue $\lambda_j$ of $\hat{O}$ as a result and make the state collapse onto the eigenvector $|\lambda_j\rangle$ of $\hat{O}$ corresponding to the obtained eigenvalue. The probability of getting the outcome $\lambda_j$ is given by $p(\lambda_j) = |\langle\psi|\lambda_j\rangle|^2$ and from Equation 2.19, the expectation value is given by $\langle\hat{O}\rangle = \sum_j p(\lambda_j)\lambda_j = \sum_j \lambda_j |\langle\psi|\lambda_j\rangle|^2$.

Now, consider the change of basis unitary $U_{\hat{O}}$, such that $U_{\hat{O}}|j\rangle = |\lambda_j\rangle$. This unitary can be easily constructed by arranging the vectors $|\lambda_j\rangle$ along the columns.

$$U_{\hat{O}} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ |\lambda_1\rangle & |\lambda_2\rangle & \cdots & |\lambda_n\rangle \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix} \tag{3.17}$$

Using this, we can write the expectation as

$$\begin{aligned} \langle\hat{O}\rangle &= \sum_j \lambda_j |\langle\psi|\lambda_j\rangle|^2 \\ &= \sum_j \lambda_j |\langle\psi| (U_{\hat{O}}|j\rangle)|^2 \\ &= \sum_j \lambda_j |\underbrace{(\langle\psi| U_{\hat{O}})}_{\langle\tilde{\psi}|}|j\rangle|^2, \end{aligned} \tag{3.18}$$

An alternative proof could be to show that the probability of getting the outcome $\lambda_j$ upon measuring $\hat{O}$ on the state $|\psi\rangle$ is the same as the probability of getting the outcome $j$ while measuring $|\tilde{\psi}\rangle$ in the computational basis. Let $|\psi\rangle = \sum_j \alpha_j |\lambda_j\rangle$. Then, $p(\lambda_j)_{|\psi\rangle} = |\alpha_j|^2$. Now, $|\tilde{\psi}\rangle = U^\dagger_{\hat{O}}|\psi\rangle = U^\dagger_{\hat{O}} \sum_j \alpha_j U_{\hat{O}}|j\rangle = \sum_j \alpha_j |j\rangle$. Hence, $p(j)_{|\tilde{\psi}\rangle} = |\alpha_j|^2 = p(\lambda_j)_{|\psi\rangle}$. where, $|\tilde{\psi}\rangle = U^\dagger_{\hat{O}}|\psi\rangle$.

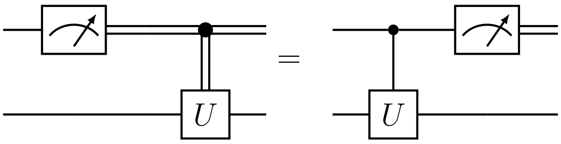
*Measuring an observable $\hat{O}$ on the state $|\psi_{out}\rangle$ is equivalent to applying the unitary $U^\dagger_{\hat{O}}$ on the state and then measuring in the computational basis $\{|j\rangle\}$.*

Note: We assume here that the observable $\hat{O}$ is non-degenerate. That is, no two eigenvectors share the same eigenvalue. This assumption is motivated by the fact that measuring an observable with degenerate eigenvalues extracts less information about the state and that is undesirable for efficiently garnering information about the output state.

For any observable, the first-order (expectation) and higher-order moments depend only on the probability distribution of the outcomes and the readout of a quantum algorithm is essentially the measurement of the probabilities $p(j)$. Similar to input state preparation, where we start off with the $|0\ldots0\rangle$ state and apply a state preparation unitary, here, we can then assume that only measurements on the computational basis are needed, provided we can apply an additional unitary $U_{\hat{O}}$ to the algorithm's output. As before, not all unitaries can be applied efficiently. Hence, again, the goal of a quantum scientist is to design algorithms where relevant information can be measured efficiently in the computational basis. That is, design observables $\hat{O}$ such that $U_{\hat{O}}$ can be realized efficiently. We now mention two subtle yet important principles, which after reading once, seem obvious.

### 3.3.1 Principle of deferred measurement

Measurements can always be deferred at the end of the circuit. If a measurement is used to control the execution of a gate on another qubit, then one may replace this by a quantum-controlled gate.



The double wire is the symbol for a classical bit. The equality above can be easily proved by comparing the outputs. This is useful as it allows us to segregate the execution and readout stages in a quantum computing architecture.

### 3.3.2 Principle of implicit measurement

Quantum systems that are not perfectly isolated often leak information to the environment through various processes (noise, dissipation, decoherence, etc). These processes might destroy quantum resources like entanglement or superposition and result in states that behave as if they are collapsed state. This process is also known as an implicit measurement, where the effect of the dissipative/decohering processes can be thought of as the effect of an (imaginary) implicit measurement. Do not worry if this is not clear right now. We shall study this in greater detail when we study density matrices.

# Bibliography

[1] John Preskill. Lecture notes for physics 229: Quantum information and computation. http://theory.caltech.edu/~preskill/ph229/, 1998. California Institute of Technology, Last accessed: September 6, 2024.

[2] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, Cambridge, 10th anniversary edition edition, 2010.

[3] Scott Aaronson. The complexity of quantum states and transformations: From quantum money to black holes, 2016.

[4] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. The Variational Quantum Eigensolver: a review of methods and best practices. *Physics Reports*, 986:1–128, November 2022. arXiv:2111.05176 [quant-ph].

[5] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, September 2021. Number: 9 Publisher: Nature Publishing Group.

[6] Dmitry A. Fedorov, Bo Peng, Niranjan Govind, and Yuri Alexeev. VQE Method: A Short Survey and Recent Developments, August 2021. arXiv:2103.08505 [quant-ph].

[7] Yudong Cao, Jonathan Romero, Jonathan P. Olson, Matthias Degroote, Peter D. Johnson, Mária Kieferová, Ian D. Kivlichan, Tim Menke, Borja Peropadre, Nicolas P. D. Sawaya, Sukin Sim, Libor Veis, and Alán Aspuru-Guzik. Quantum Chemistry in the Age of Quantum Computing. *Chemical Reviews*, 119(19):10856–10915, October 2019. arXiv:1812.09976 [quant-ph].

[8] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qute.201900070.

[9] Han Qi, Sihui Xiao, Zhuo Liu, Changqing Gong, and Abdullah Gani. Variational quantum algorithms: fundamental concepts, applications and challenges. *Quantum Information Processing*, 23(6):224, June 2024.

[10] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92(1):015003, March 2020. Publisher: American Physical Society.

# Chapter 4

# Quantum Algorithms

## 4.1 Quantum algorithms and quantum advantage

Quantum algorithms, like classical algorithms are a series of quantum operations (circuits) that have been designed to solve a particular task efficiently. As mentioned in the previous chapter, we leverage quantum effects like entanglement and interference to encode and decode the information we need, with a higher efficiency than on a classical computer. However, it is very important to keep in mind that coming up with such schemes is a fundamentally hard task and sometimes requires decades of research, of course, with a (sizable) bit of good luck!

We will now study the first two instances of problems where quantum advantages have been demonstrated. That is, problems which are designed to be difficult for a classical computer to solve with sub-exponential complexity, but are easy to solve on a quantum computer.

## 4.2 The Deutsch algorithm

The Deutsch algorithm is the first quantum algorithm that exemplifies the quantum speedup over a classical computational task. Published in 1985, it worked as the birth of quantum computing as an applied domain of research in computer science. It also illustrates the idea of quantum parallelism and how it can be leveraged to gain a quantum advantage.

**Problem statement:** Given a function $f : \{0, 1\} \to \{0, 1\}$ which maps 1 bit to 1 bit, we want to determine if $f(0) = f(1)$.

**Classical solution:** To classically solve this problem, we would need to evaluate the function $f$ at least 2 times.

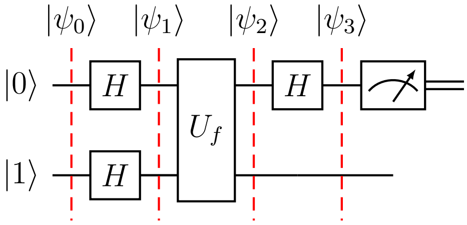
**Quantum solution:** We can answer this question by evaluating the function $f$ only once but on a quantum computer.

Consider we have a quantum "oracle" that applies $f(x)$, namely, it applies a two-qubit (reversible) unitary:

$$U_f : |x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle \tag{4.1}$$

where $\oplus$ is sum modulo 2. That is, $1 \oplus 1 = 0 \oplus 0 = 0$ and $1 \oplus 0 = 0 \oplus 1 = 1$.

Hence, $U_f$ flips the bit $y$ if $f(x) = 1$. Further, (obviously) it acts linearly on quantum superpositions. The quantum circuit of the algorithm is given below.



Let's compute the wave functions.

$$\begin{aligned}
|\psi_0\rangle &= |01\rangle \\
|\psi_1\rangle &= \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \\
|\psi_2\rangle &= \frac{1}{2}((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle)(|0\rangle - |1\rangle) \\
|\psi_3\rangle &= \frac{1}{2\sqrt{2}}\left\{\left((-1)^{f(0)} + (-1)^{f(1)}\right)|0\rangle + \left((-1)^{f(0)} - (-1)^{f(1)}\right)|1\rangle\right\}(|0\rangle - |1\rangle)
\end{aligned} \tag{4.2}$$

Upon measuring the first qubit, we get,

$$\begin{aligned}
f(0) = f(1) &\implies p(0) = 1, p(1) = 0 \\
f(0) \neq f(1) &\implies p(0) = 1, p(1) = 1.
\end{aligned} \tag{4.3}$$

Hence, just by measuring the first qubit we can tell if the function $f$ is balanced or not for the outcomes 0 and 1 respectively. Therefore, we have solved the problem by evaluating the function $f$, through the oracle $U_f$, only once!

**How is this possible?** We achived this through a two step process.

1. The first layer of Hadamard gates created a superposition of the four possible 2 qubit basis states, namely, $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$. This ensures that all the states are evaluated in one go and is hence known as *quantum parallelism*. However, this does not guarantee that we can extract all this information freely.

2. For extracting the encoded information optimally, we use a technique called *phase kickback* along with the second Hadamard gate on the control qubit. This is the tricky part–using quantum effects (interference) and clever measurements to ensure that the result of the collapse helps us gain the information we need.

**Phase kickback:** Consider the action of $U_f$ on $|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle|-\rangle + |1\rangle|-\rangle)$

$$\begin{aligned}
|00\rangle - |01\rangle &\xrightarrow{U_f} |0f(0)\rangle - |0\overline{f(0)}\rangle = |0\rangle(-1)^{f(0)}|-\rangle \\
|10\rangle - |11\rangle &\xrightarrow{U_f} |1f(1)\rangle - |1\overline{f(1)}\rangle = |1\rangle(-1)^{f(1)}|-\rangle
\end{aligned} \tag{4.4}$$

Since these phases are a global phases on the second qubit in state $|-\rangle$, they can be *kicked back* onto the first qubit. Then, when we consider the joint state of the two qubits, these phases introduce a relative phase in $|\psi_2\rangle$. This phenomenon of kicking back the phase is known as phase kickback.

The second H gate then leverages this relative phase to modify the amplitudes of the first qubit. The H gate is a unitary transform between the $Z$ basis $|0\rangle, |1\rangle$ to the $X$ basis $|+\rangle, |-\rangle$. When we apply the second H gate, we go from having indiviudal phases on $|0\rangle$ and $|1\rangle$, to having the (normalized) sum and the difference of these phases on $|0\rangle$ and $|1\rangle$, respectively. This is precisely because the H gate transforms the states $|0\rangle$ and $|1\rangle$, to the (normalized) sum and the difference of these states.

The algorithm works owing to **(a)** quantum parallelism by preparing a superpositon of states with the required information, **(b)** evaluating the objective function through the oracle, **(c)** phase kickback that can be encoded as a relative phase in the computational basis and gives rise to interference **(d)** which is later leveraged for extracting the required information from the measurement amplitudes. The beauty of the Deutsch's algorithm is that it already contains all the ingredients that build up most of the more complex quantum algorithms.

More importantly, it highlights the difference between quantum parallelism and classical randomized algorithms. We might be tempted to interpret $|0f(0)\rangle + |1f(1)\rangle$ as a probabilistic classical computation where with equal probability we either compute $f(0)$ or $f(1)$. This is not true since the two events on a classical computer will be mutually exclusive. However, on a quantum computer, these two amplitudes interfere and hence give us a more efficient way to extract the information we need. This is an excellent example of what we emphasised in the previous lecture.

It may seem that we have indeed been lucky with the simplicity/structure of the problem and that it may be harder to encode and then extract information from a superposition for an arbitrary problem. And yes, that statement is partly true. We will soon see a generalized quantum Fourier Transform which will enable us to design more general algorithms and encompass broader classes of problems.

Deutsch's algorithm is very simple and, while it shows the principle of quantum speedup, it does not say anything about the scaling of the speedup with the size of the problem (it has fixed size). For this, there were three historical algorithms that demonstrated several principles: the Deutsch-Jozsa (1992), Bernstein-Vazirani (1997), and Simon's algorithm (1997). We now investigate the first in this lecture and then explain the other two after a brief discussion on quantum complexity in the next lecture.

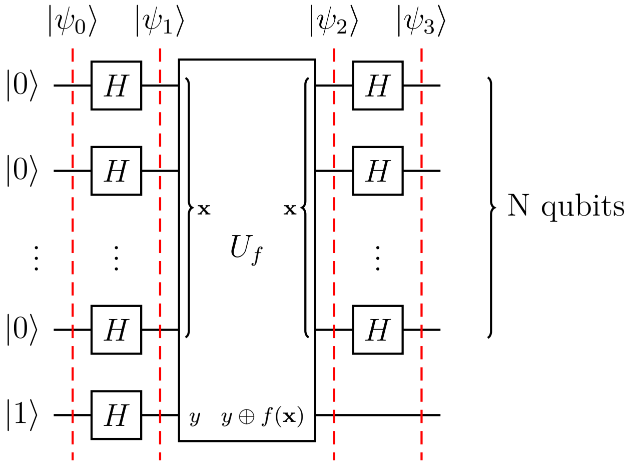
## 4.3 The Deutsch-Jozsa algorithm

The Deutsch-Jozsa algorithm is an extension of the Deutsch algorithm to multiple qubits, consequently demonstrating the advantage with scaling the size of the algorithm.

**Problem statement:** Given a "black-box" function $f(\mathbf{x})$, such that, $f : \{0,1\}^n \to \{0,1\}$ the task is to decide if it is constant or balanced. Here, constant means $f(\mathbf{x}) = 0 \;\forall\; \mathbf{x}$ or $f(\mathbf{x}) = 1 \;\forall\; \mathbf{x}$, while balanced means that there are $2^{n-1}$ values of $\mathbf{x}$ for which $f(\mathbf{x}) = 0$ and $2^{n-1}$ values for which $f(\mathbf{x}) = 1$. The problem is the natural generalization of Deutsch's algorithm.

**Classical solution:** To classically solve this problem, we could try to be a bit smart and play our luck by randomly selecting values of $\mathbf{x}$, and if $f(x_i) \neq f(x_j)$ for any two trials $i, j$, then we can safely conclude that $f$ is balanced. However, in the worst-case scenario, to determine the answer with complete uncertainty, we would need to evaluate the function $2^{n-1}+1$ times.

**Quantum solution:** The quantum circuit is also a natural generalization of the circuit for Deutsch's algorithm.



The principle is the same:

$$|\psi_0\rangle = |01\rangle$$

$$|\psi_1\rangle = \left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|\mathbf{x}\rangle\right)\otimes\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right). \tag{4.5}$$

Here, $|\mathbf{x}\rangle\ \forall\, x\in\{0,1\}^n$ is the quantum state on $n$ qubits, where each qubit is in the state $|0\rangle$ or $|1\rangle$, depending on the corresponding bit in the binary representation of $\mathbf{x}$.

*Exercise:* Show that $H^{\otimes n}$ transforms $|0\rangle^{\otimes n}$ to $\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle$.

*Hint:* We can notice a pattern by starting with $n=1,2$.

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}}\sum_x(-1)^{f(x)}|x\rangle\otimes\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \tag{4.6}$$

Then, the oracle unitary $U_f$ gives rise to the phase kickback. To compute $|\psi_3\rangle$, we first write the action of H on a single qubit state $|y\rangle\,; y=0,1$, in a more concise notation.

$$H|y\rangle = \frac{1}{\sqrt{2}}\sum_{z=0}^{1}(-1)^{yz}|z\rangle \tag{4.7}$$

Now, in tha case of $N$ qubits, for $|\mathbf{x}\rangle = |x_1x_2\ldots x_n\rangle$, we get,

$$H^{\otimes n}|x_1...x_n\rangle = \frac{1}{\sqrt{2^n}}\sum_{z_1...z_n\in\{0,1\}^n}(-1)^{x_1z_1+...+x_nz_n}|z_1...z_n\rangle\,. \tag{4.8}$$

We introduce the bit-wise inner product modulo 2 as $\mathbf{x}.\mathbf{z} = x_1 z_1 + ... + x_n z_n (\text{mod} 2)$ and hence can write

$$H^{\otimes n} |\mathbf{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{z}} (-1)^{\mathbf{x}.\mathbf{z}} |\mathbf{z}\rangle . \tag{4.9}$$

Combining the above few results, we can obtain $|\psi_3\rangle$ as,

$$|\psi_3\rangle = \left( \frac{1}{2^n} \sum_{\mathbf{x},\mathbf{z} \in \{0,1\}^n} (-1)^{\mathbf{x}.\mathbf{z}+f(\mathbf{x})} |\mathbf{z}\rangle \right) \otimes \left( \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) . \tag{4.10}$$

Let us focus on the amplitude of the $|\mathbf{z} = 0\rangle$ component, given by $\langle \mathbf{x} = 0 | \psi_3 \rangle$.

$$\frac{1}{2^n} \sum_{\mathbf{x}} (-1)^{f(\mathbf{x})} = \begin{cases} 0 & \text{if } f \text{ is balanced.} \\ \pm 1 & \text{if } f \text{ is constant} = 0 \text{ or } 1 \text{ respectively } \forall\, \mathbf{x}. \end{cases} \tag{4.11}$$

The probability of measuring $|\mathbf{z} = 0\rangle$ upon measurement is given by the modulus squared of the amplitude $|\langle \mathbf{z} = 0 | \psi_3 \rangle|^2$. If $f$ is balanced, then the probability of measuring $|\mathbf{z} = 0\rangle$ is exactly 0 since half the terms will each contribute a $(-1)^0 = +1$ and the other half will each contribute $(-1)^1 = -1$ to the sum, hence nullifying the total amplitude and probability.

On the other hand, if $f$ is constant, then the each term contributes $\pm 1$ depending on the constant value of $f$, 0 or 1, respectively. Nonetheless, for both types of constant functions, the probability of measuring $|\mathbf{z} = 0\rangle$ is 1. Therefore, just by performing, one measurement, we can solve the problem with complete certainty.

Note: In this lecture, we have assumed the oracle takes the same time to execute independent of $n$. That is, the complexity of the oracle is $O(1)$. Hence, the quantum algorithm runs in a time order of what it takes to compute $f(\mathbf{x})$ once. This assumption on the oracle is very strong and absolutely non-trivial in general. This is why sometimes such speedups are called "relativized" speedups because they are obtained with the help of an oracle. Naturally, this opens up another research direction focused on developing efficient oracles for a given problem. The best example of such a problem, will be the factorisation problem which we shall discuss in the upcoming lectures.

*Most quantum algorithms assume an efficient oracle and making their advantage relativised. Therefore, in addition to cleverly leveraging quantum effects, we also need to focus on efficient oracle realisations. This is one of the many gentle reminders of the non-universality of quantum computing.*

In the beginning, we mentioned that the worst-case complexity for deterministically solving the problem is to evaluate the function $2^{n-1} + 1$ times. However, since quantum algorithms are, in general, probabilistic, it makes sense to compare them with classical probabilistic algorithms. These are algorithms that give us the answer with some confidence $1 - \epsilon$. In our case, we may content ourselves with knowing whether $f$ is balanced or constant with probability $\mathbb{P} = 1 - \epsilon$ with $\epsilon$ small.

If the function is balanced, then the probability of getting the same answer for $k$ queries is given by $p = 1/2^{k-1}$. If this happens, then we may wrongly guess that $f$ is constant. Hence, the probability of us guessing wrong is $p$. Now, we want the error rate to be

$p \leq \epsilon \implies 2^{-k+1} \leq \epsilon \implies 2^{k-1} \geq \frac{1}{\epsilon} \implies k \geq 1 + \log\left(\frac{1}{\epsilon}\right)$. Hence, we can decrease the error rate exponentially with the number of queries $k$, we make. Hence, the task is not exponentially hard classically.

Deutsch-Jozsa algorithm demonstrates a speedup, but only polynomial in $n$ for fixed $\epsilon$, that too with access to an oracle, and only in the worst case scenario. Can we relax any of these restrictions? Yes, partly. Thanks to two quantum algorithms developed in 1997, the Bernstein-Vazirani and Simon's algorithm. We shall first study a bit of classical and quantum computational complexity theory, which will help us better appreciate the advantages introduced by Bernstein-Vazirani and Simon's algorithms in the next lecture.

# Chapter 5

# Computational Complexity

Having introduced the two foundational quantum algorithms, we will now explore the broader landscape of classical and quantum complexity. This will set the stage for studying two additional algorithms that offer even more significant quantum advantages. The aim of this lecture is to provide a high-level understanding of the complexity framework and key insights behind these algorithms without delving into all the technical details. It is perfectly fine if some concepts feel abstract at this stage—they will become clearer as we study additional algorithms in more depth later on.

## 5.1 Computational complexity

Computational complexity (classical or quantum) is the study of the inherent difficulty of solving a given problem. The complexity is defined with regards to how much resource is required by an algorithm that solves the problem. We shall begin by discussing classical computational complexity and then move onto quantum computational complexity.

### 5.1.1 Classical computational complexity

Any classical problem is associated with two kinds of complexities[1].

- **Time complexity:** The time taken by the algorithm to run. This scales directly with the number of operations performed by the algorithm.
- **Space complexity:** The space or memory consumed by the algorithm.

There exists an *inevitable trade-off* between the two. That is, if we were to reduce the time complexity of an algorithm, we would inevitably increase its space complexity. In these lectures, we shall mainly focus on studying the time complexity and the resulting complexity classes in which we can classify problems.

Before we begin, to ensure a fair comparison, it is crucial to mention that any problem can be modeled as an equivalent decision problem. A decision problem is a yes/no question on a plausibly infinite set of inputs. This equivalence is the result of years of work by

[1] These complexities are usually considered with respect to the best algorithm that solves the problem.

Turing, Cook, Levin and Karp[2]; which for good reason, shall be skipped in these lectures. Nonetheless, to convince ourselves, we demonstrate the same through a few examples.

- **Search:** Search for an element $x^*$ in a set $S$.
  *Equivalent decision problem:* For any element $y \in S$, ask if $y = x^*$?
- **Optimization:** Find the value $x^*$ for which the function $f(x)$ is minimal.
  *Equivalent decision problem:* For each input $x$, ask if there exists an element $y$ such that $f(y) < f(x)$?
- **Sudoku:** The objective is to solve a Sudoku puzzle.
  *Equivalent decision problem:* For the current state of the $n \times n$ grid, does there exist a set of numbers which, when filled in the blanks, completes the puzzle?
- **Chess:** The objective is to win a game of chess in some number of moves $k$.
  *Equivalent decision problem:* Given the current state of the game, can one side (say White) guarantee a win in the next $k$ moves?

## 5.2 Classical deterministic complexity classes

We define the complexity classes with regard to the time complexity, equivalently, the number of operations or decisions we need to make to reach the solution. That is, "*How many yes/no questions do we need to answer before finding a solution?*" Further,

- Do we want to find the solution or simply verify a given candidate solution?
- Do we want an answer with absolute certainty $p = 1$ or are we fine with a probabilistic answer with $p = 1 - \varepsilon$?
- Do we want our algorithm to be classical or quantum?

Based on these criteria, we motivate the different classical and quantum complexity classes. The classes we state below are with regard to the current best algorithms for solving these problems. It is quite plausible that a problem in a certain class might be shifted to a different class in the future upon discovering a more efficient algorithm.

### 5.2.1 Tractability

Tractable problems are problems for which there exists an algorithm that provides a solution in $O(\text{poly}(n))$ time. That is, the time it takes to solve such problems scales polynomially with the input size $n$.

*Example:* Searching for an element in a $n$ element sorted array takes $O(\log n)$ time.

Intractable problems on the other hand are problems for which the best algorithm has an exponential run-time $O(\exp(n))$.

*Example:* The travelling salesman problem. Given $n$ cities and the distance between them, we need to find a route such that we visit all the cities but cover the minimum possible distance. The best algorithm in the general case is the Held-Karp algorithm with a complexity $O(n^2 2^n)$.

[2] Please see the Wikipedia pages on Decision Problems and the Cook-Levin Theorem. If you would like to discuss these topics further or need more resources, please feel free to reach out :)

Sub-exponential or quasi-polynomial problems are problems that can be solved faster than exponential time but slower than polynomial time. These exist in a grey area between tractable and intractable problems and the choice to include them in either of the classes is usually nuanced with good reason.

*Example:* Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $V_1$ and $V_2$ are sets of vertices and $E_1$ and $E_2$ are sets of edges, determine if there exists a bijection $f : V_1 \to V_2$ such that for every pair of vertices $u, v \in V_1$, $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$. In other words, $G_1$ and $G_2$ are said to be isomorphic if there exists a one-to-one mapping of vertices $f : V_1 \to V_2$ such that $G_1$ and $G_2$ have identical edge structures. The best algorithm is Babai's from 2015 and has a quasi-polynomial complexity $O(2^{c \log n})$, where $c$ is a constant.

### 5.2.2 P

Polynomial ($P$) is the set of all problems for which we can find a solution using a polynomial time $O(\text{poly}(n))$ algorithm. This is the same set of problems as tractable problems.

### 5.2.3 NP

Non-deterministic polynomial ($NP$) is the set of problems where if we are given a candidate input, then we can verify if this candidate is truly a solution or not, in polynomial $O(\text{poly}(n))$ time.

*Integer factorization:* Given an integer $M$ (with $n$ bits in its binary representation), find two non-trivial ($> 1$) integer factors $k_1$ and $k_2$, such that $M = k_1 k_2$. If we were to brute force the task by searching for factors starting from 2, then the complexity would be $O(2^n)$, where in the worst case, $M = 2^n - 1$ is prime and it takes us $O(2^n)$ tries to figure that out. If we act smarter and only check the odd numbers, even then, the complexity would be $O(2^{n/2})$. Nonetheless, given two candidates $k_1$ and $k_2$, we can easily verify if their product is equal to $M$ or not in polynomial time.

Naturally, all problems where we can find the true solution in polynomial time (hence are automatically verifiable), also belong to this set, that is, $P \subset NP$, as shown in Figure 5.1.

Further, there are some arguments that $NP$ problems can be solved in polynomial time using a truly random Turing machine–a computer that explores all possible paths simultaneously at each step and halts when it finds the solution. However, having such a machine is far from reality as explained in **??**, and hence it is conjectured whether $P \overset{?}{=} NP$, and we would like to believe NO! Some of us might be wondering *Why are they called non-deterministic problems?* We provide a brief answer for the same in **??**, but skip it for now since it is not required.

### 5.2.4 NP-Complete

NP-Complete ($NPC$) is a special subset of $NP$ problems. The speciality being that we can reformulate/map any $NP$ problem as an $NPC$ problem. Since $NPC$ problems are also $NP$ problems, it means that all $NPC$ problems can be reformulated as one another.

*Boolean satisfiability:* The first and most general example of the same is the boolean satisfiability or $n$-SAT problem. The general $n$-SAT problem is any possible boolean

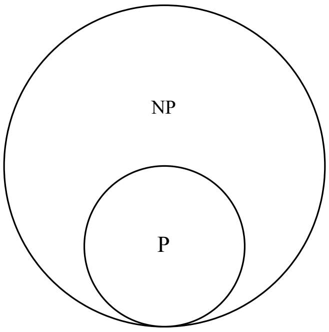


Figure 5.1: Illustration of $P \subset NP$.

expression with $n$ boolean variables. For example, a 3-SAT problem can be $f(\mathbf{x}) = (x_1 \wedge x_2) \vee (x_2 \wedge \overline{x}_3)$, where $x_i \in \{0, 1\}\ \forall\ i = 1, 2, 3$, the $\overline{x}_i$ represents the negation operation $0 \leftrightarrow 1$, the $\wedge$ represents the AND operation and the $\vee$ is the OR operation. The Cook-Levin theorem states that any other $NP$ and even $P$ (since $P \subset NP$) problem can be reformulated/mapped to an $n$-SAT problem.

*Claim:* Although $NPC$ is a special subset of $NP$ problems ($NPC \subset NP$), solving any NPC problem would mean we can solve all NP problems.

- Firstly, if we have an algorithm to solve one NPC problem, we can solve all of them, since all NPC problems can be mapped to one another.
- Further, since we can map any NP problem onto an NPC problem, this would also mean, we can use our algorithm to solve any NP problem.

Hence, we have now established two relations. Firstly, $P \subset NP$ and now $NPC \subset NP$. However, $NPC$ and $P$ are not the same. In fact, the problems in $NPC$ are $NP$ problems, and hence cannot be solved in polynomial time, but rather only candidate inputs can be verified in polynomial time, as shown in Figure 5.2.

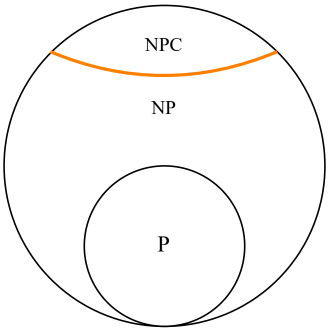


Figure 5.2: Illustration of $P \subset NP$ and $NPC \subset NP$.

### 5.2.5 NP-Hard

NP-Hard ($NPH$) is a set of problems made up of two classes: **(a)** $NPC$ problems and **(b)** problems that are beyond $NP$. These are problems that cannot be solved, nor can we verify if a candidate input is a true solution, within polynomial time.

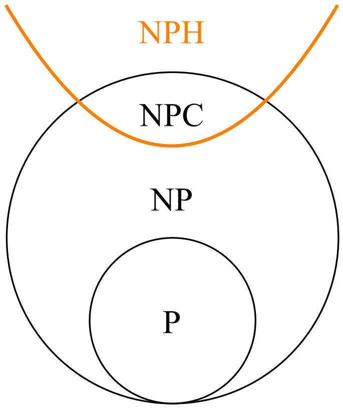


Figure 5.3: An overview of the deterministic classical complexity classes.

*Examples:* Two main examples of problems outside $NP$ are a game of chess and the Halting problem. The latter is truly a pathological problem that is impossible to solve.

This concludes the discussion on the classical deterministic computational complexity, and we present an overview of the same in Figure 5.3. Now, we shall look at probabilistic complexity classes. Where, we relax the constraint of finding or verifying solutions with complete certainty.

## 5.3 Probabilistic computational complexity classes

Before jumping into the probabilistic complexity classes, lets briefly recap $P$ and $NP$. $P$ is the set of problems whose solution can be found in polynomial time, whereas $NP$ is the set of problems where we can only verify if a given input is a solution or not, in polynomial time. However, the process of finding/proposing the candidate may or may not run in polynomial time or we might have to propose exponentially many such candidates. This is where the probabilistic algorithms step in.

### 5.3.1 BPP

Bounded error probabilistic polynomial $BPP$ is the set of problems for which we can use a polynomial time algorithm to predict the correct candidate with a success probability $p > 2/3$. This set $BPP$ is related to both $P$ and $NP$.

- $BPP$ encompasses the set $P$, that is, $P \subset BPP$ since $p = 1$ for all problems in $P$.
- $BPP$ is simply the subset of $NP$ problems for which we can predict the correct answer (with probability $p > 2/3$) using a polynomial time algorithm. At the same time, not all $NP$ problems can be solved probabilistically with a success probability $> 2/3$, in polynomial time. Hence, $BPP \subset NP$ but not the other way round. This gives us $P \subset BPP \subset NP$, as shown in Figure 5.4.

Running a $BPP$ algorithm $T$ times would result in an error probability of $(1-p)^T$, which goes to 0, exponentially fast with $T$. However, for any finite $T$, there still exists a (small but non-zero) probability of getting the wrong answer. This is an important distinction between $BPP$ and $P$, where in the latter, we are sure to obtain the true solution with a finite (in fact polynomially large) number of operations.

Interestingly, the number of problems that are in $BPP$ but not in $P$ is decreasing due to the process of *de-randomization*. Where, we take a polynomial-time probabilistic algorithm that solves some problem with a success probability $p > 2/3$, and develop a deterministic version of the same. Hence solving the problem with complete certainty, in polynomial time. This process is highly non-trivial and often takes years if not decades to complete. These advancements have led to the conjecture $P \overset{?}{=} BPP$[3], which again, we would like to believe is not true!.

*Primality test:* Given an $n$-bit integer $M$, the task is to determine if it is prime or not. There existed no polynomial time deterministic algorithm before Agrawal–Kayal–Saxena's deterministic algorithm (2002).

*Polynomial identity testing:* One of the problems that has not been de-randomized yet, is the polynomial identity testing problem. Given two polynomials in $n$ variables $f_1(x_1, x_2, \ldots, x_n)$ and $f_1(x_1, x_2, \ldots, x_n)$, the problem is to find whether $f_1$ and $f_2$ are the same polynomial. Equivalently, we need to check if their difference is zero, that is, $f_1(x_1, x_2, \ldots, x_n) - f_1(x_1, x_2, \ldots, x_n) = 0 \ \forall \ x_i \ \forall \ i = 1, 2, \ldots, n$. The Schwartz–Zippel algorithm solves this problem in polynomial time with a success probability $p > 2/3$.

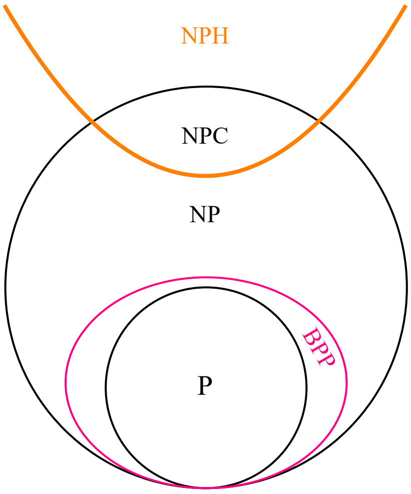


Figure 5.4: $P \subset BPP \subset NP$.

### 5.3.2 MA

Building upon the $BPP$ set, Merlin-Arthur ($MA$) is the set of problems where, we can solve it using a Merlin-Arthur strategy. Here, Merlin provides a candidate input and Arthur uses a polynomial time probabilistic algorithm to verify if the given candidate is truly a solution or not. Arthur's verifier needs to be correct at least two-thirds of the time.

[3] Please see **??** for more details.

1. If Merlin provides a true solution, then Arthur's verification algorithm should be able to verify that indeed it is solution at least two-thirds of the time, that is, with a probability $p_1 > 2/3$.
2. If Merlin provide a false candidate, then Arthur's verification algorithm should mistake it to be a true solution not more than two-thirds of the time, that is, with a probability $p_2 < 1/3$. Equivalently, it should spot out the false candidates at least two-thirds of the time, that is, with a probability $1 - p_2 > 2/3$.

Just like $BPP$ randomizes the process of finding the candidate, $MA$, takes this a step further and makes the verification process also probabilistic.

- $BPP \subset MA$. If a problem can be solved using a $BPP$ algorithm, then Arthur's verification procedure is to simply run the $BPP$ algorithm, regardless of Merlin's input. Hence, all $BPP$ problems can be solved using a trivial $MA$ algorithm.
- $NP \subset MA$. Similar to $BPP$ being the probabilistic version of $P$, that is, $P \subset BPP$, with $p = 1$, here, $MA$ is the probabilistic version of $NP$. That is $NP \subset MA$, with Arthur's verification process having a probability of success $p_1 = 1$.

Finally, analogous to the $P \stackrel{?}{=} NP$ conjecture, we also have $BPP \stackrel{?}{=} MA$. And under the conjecture of de-randomization, we have $NP \stackrel{?}{=} MA$.

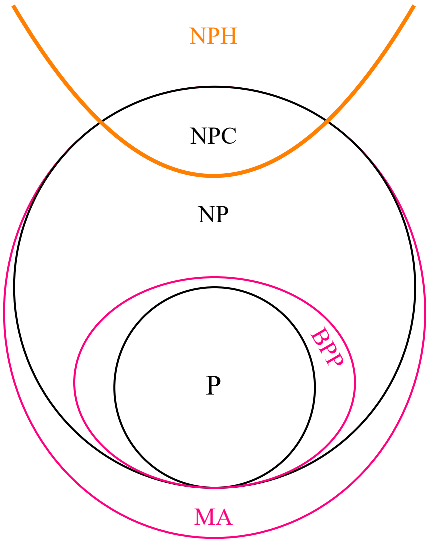


Figure 5.5: The complete hierarchy involving deterministic and probabilistic classical comptational complexity classes.

### 5.3.3 Summary

As a recap, we go over the classes briefly. $P$ is the set of all polynomial-time solvable problems and $BPP$ is the set of such problems but with a success probability $p > 2/3$. $NP$ is the set of all polynomial-time verifiable problems and $MA$ is the set of such problems but the verifier (Arthur) succeeds with a probability $p_1 > 2/3$. Additionally, in the deterministic case, there is a special subset of $NP$ problems, called $NPC$, which are problems onto which we can map all $NP$ problems. This special subset, plus all the problems that are not in $NP$, make up the $NPH$ set.

*We have covered two kinds of conjectures.*
*Regarding hardness* $P \stackrel{?}{=} NP$, $BPP \stackrel{?}{=} MA$.
*Regarding de-randomization* $P \stackrel{?}{=} BPP$, $NP \stackrel{?}{=} MA$.

## 5.4 Quantum Computational Complexity

Quantum computational complexity involves the characterisation of problems based on the quantum resources required to solve them. We can define time and space complexity in a very similar manner as in the classical case. Like the classical complexity classes, we can also define quantum complexity classes, with regards to the time complexity of the quantum algorithms that solve these problems. Additionally, we shall also highlight the relations between classical and quantum complexity classes.

### 5.4.1 BQP

Analogous to $BPP$, the bounded error quantum probability time ($BQP$) is the set of all problems that can be solved using a quantum computer in polynomial time, with a success probability of $p > 2/3$. Since classical computations are a subset of quantum computations, $P \subset BPP \subset BQP$, as shown in Figure 5.6. Needless to say, $BQP \subset NP$ since not all $NP$ problems can be solved with a good success probability ($p > 2/3$) in polynomial time on a quantum computer.

*Quantum integer factorization:* Shor's algorithm solves the integer factoring problem in $O((\log N)^2 \cdot \log\log N)$, with a success probability $p > 2/3$.

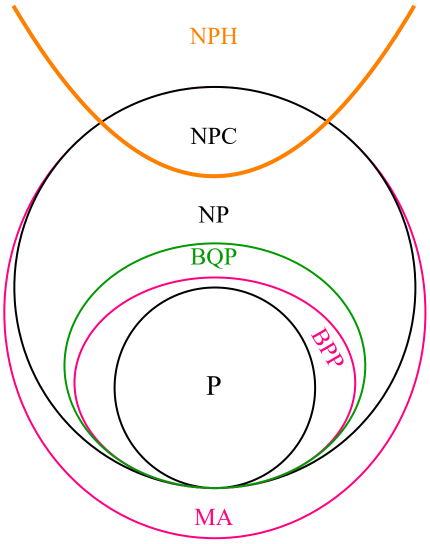


Figure 5.6: The quantum complexity class $BQP$ encompasses $BPP$ which encompasses $P$. However, they are all subsets of $NP$.

### 5.4.2 QMA

Finally, we introduce $QMA$, the set of problems where Merlin again provides a candidate input (quantum state) and Arthur uses a polynomial time *quantum* algorithm to verify if the given state is a solution, with the following probabilities.

1. If Merlin provides a solution state, then Arthur's algorithm should be able to verify that indeed it is solution, at least two-thirds of the times, that is, with a probability $p_1 > 2/3$.
2. If Merlin provide a false state, then Arthur's verification algorithm should mistake it to be a true solution no more than one-third of the time, that is, with a probability $p_2 < 1/3$.

It is natural to see that this is simply an extension of $MA$, where Arthur now has a polynomial time quantum verification algorithm instead of a classical one. This completes our overview on quantum and classical computational complexity. We present a graphical representation of the same in Figure 5.7.

*Local Hamiltonian problem:* Given a $k$-local Hamiltonian, the task is to find its ground state. A $k$-local Hamiltonian is a Hamiltonian $\hat{H} = \sum_i \hat{H}_i$, where each $\hat{H}_i$ acts on at-most $k$ qubits. This is a problem which belongs to $QMA$ [4].

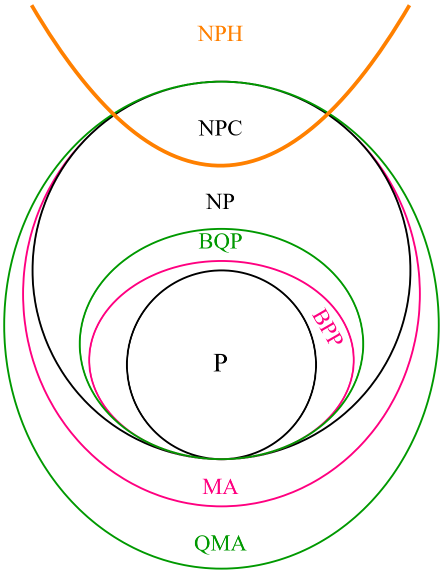


Figure 5.7: A graphical overview of the quantum and classical computational complexity landscape.

## 5.5 Oracle separation

While studying the quantum algorithms in the previous lecture (Deutsch and Deutsch-Jozsa), assumed the existence of an *oracle*. Even while defining the quantum computation complexity classes above, we implicitly made the same assumption.

[4] Try to think why, or wait a few lectures till we study quantum phase estimation :)

**Orcale:** An oracle is a black-box that gives us the value of a function $f(x)$ for any input $x$, within polynomial time.

The ability of quantum algorithms to perform better than classical algorithms while both are using an oracle, is known as oracle-relativized advantage or an *oracle-separation.* This is by no means an unfair advantage, as both computers have access to the same system. In fact, this framework establishes a fair basis for comparing classical and quantum algorithms.

The first few algorithms to demonstrate an oracle separation between the classes $BPP$ and $BQP$ were the Deutsch-Jozsa, Bernstein-Vazirani and Simon's algorithm. Deutsch-Jozsa, although being the first, demonstrates an exponential quantum advantage but only in the (a) worst case scenario of the problem and (b) the classical algorithm is strictly deterministic. Bernstein and Vazirani helped drop the first condition and then Simon helped drop both the conditions.

### 5.5.1 Bernstein-Vazirani

The Bernstein-Vazirani algorithm (1993) erradicates the need for the first condition and shows a quasi-polynomial quantum advantage even over the classical algorithm in the best possible case of the problem.

**Problem statement:** We are given a function $f : \{0,1\}^n \to \{0,1\}$, which computes the modulo 2 dot product between the input and a hidden binary string $\mathbf{r}$. That is, $f(\mathbf{x}) = x_1 \oplus r_1 + x_2 \oplus r_2 + \cdots + x_n \oplus r_n \pmod 2$. Our task is to find the hidden bit-string $\mathbf{r}$. Both classical and quantum computers have access to the oracle which outputs $f(\mathbf{x})$ for each input $\mathbf{x}$ in polynomial time.

**Classical solution:** The best possible classical solution would require $n$ queries to the oracle, namely, the strings $\{100\ldots0, 010\ldots0, \ldots 00\ldots1\}$, each would help us find one bit from the string $\mathbf{r}$.

**Quantum solution:** The quantum solution only requires 1 query to determine the hidden string $\mathbf{r}$.

This algorithm is in very similar spirit as the Deutsch-Jozsa algorithm but here, we show an advantage even in the best case scenario of the problem as opposed to only the worst case scenario in Deutsch-Jozsa. However, they both demonstrate an advantage only over deterministic classical algorithms. Simon's algorithm demonstrates an exponential advantage with respect to the best possible probabilistic classical algorithm.

### 5.5.2 Simon's Algorithm

Simon's algorithm was the first quantum algorithm to show an exponential advantage over the best possible classical probabilistic algorithm, even in the best case scenario.

**Problem statement:** We are given a function $f : \{0,1\}^n \to \{0,1\}^n$ with the promise that there exists a secret $n$-bit string $\mathbf{r}$, such that for every pair of distinct inputs $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$, $f(\mathbf{x}) = f(\mathbf{y})$ if and only if $\mathbf{y} = \mathbf{x} \oplus \mathbf{r}$. In other words, $f$ is a two-to-one function such that $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{r})$ for a hidden string $\mathbf{r}$ and outputs distinct values otherwise. Our goal is to determine the hidden string $\mathbf{r}$.

**Classical solution:** The best classical probabilistic algorithm requires $\mathcal{O}(2^{n/2})$ queries to the oracle to find $\mathbf{r}$ with high probability. This is because, to detect the hidden periodicity, a classical algorithm needs to find at least two distinct inputs $\mathbf{x}, \mathbf{y}$ such that $f(\mathbf{x}) = f(\mathbf{y})$, which would require searching through a significant portion of the input space.

**Quantum solution:** Simon's quantum algorithm can find the hidden string $\mathbf{r}$ using only $\mathcal{O}(n)$ queries. It leverages quantum parallelism and interference to extract linear equations for $\mathbf{r}$, which can then be solved classically in polynomial time to identify the string with high probability.

Simon's algorithm demonstrates an exponential advantage over the best possible classical probabilistic algorithm, even in the best-case scenario. This result was historically significant as it provided one of the first examples of an exponential oracle separation, highlighting the superiority of quantum algorithms even over the best classical probabilistic approaches. This breakthrough paved the way for revolutionary algorithms like Shor's, which we shall study in detail soon.

# Chapter 6

# Quantum Fourier Space

## 6.1 The quantum Fourier transform

Being a quantum operation, the QFT is linear, and hence, it is sufficient to study the effect of the QFT on the basis states. Consider an $n$-qubit basis state $|j\rangle$, where $0 \leq j \leq 2^n - 1$ is the integer represented by the $n$ bit binary bit-string representation of the $n$-qubit basis. The QFT is then defined as

$$\boxed{\mathrm{QFT}\,|j\rangle = \frac{1}{\sqrt{2^n}} \sum_{\ell=0}^{2^n-1} e^{ij\ell\frac{2\pi}{n}}\,|\ell\rangle = \frac{1}{\sqrt{N}} \sum_{\ell=0}^{N-1} \omega_N^{j\ell}\,|\ell\rangle\,,} \tag{6.1}$$

where, $N = 2^n$ and $\omega_N = e^{i\frac{2\pi}{n}}$ is the $n^{\text{th}}$ root of unity. Equivalently, we can define the $2^n \times 2^n$ large matrix representation $(F)$ as

$$F = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \dots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \dots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \dots & \omega_N^{(N-1)(N-1)} \end{bmatrix}. \tag{6.2}$$

We can naturally study the effect of applying the QFT to an arbitrary quantum state $|\psi\rangle = \sum_{j=0}^{2^n-1} c_j\,|j\rangle$,

$$\begin{aligned} |\varphi\rangle = \mathrm{QFT}\,|\psi\rangle = \mathrm{QFT}\left(\sum_{j=0}^{2^n-1} c_j\,|j\rangle\right) &= \sum_{j=0}^{2^n-1} c_j\,\mathrm{QFT}\,|j\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} c_j \sum_{l=0}^{2^n-1} \omega_N^{j\ell}\,|\ell\rangle\,, \end{aligned} \tag{6.3}$$

where the $\ell^{\text{th}}$ component of the output state $|\varphi\rangle = \sum_{\ell=0}^{2^n-1} d_\ell\,|\ell\rangle$ is

$$d_\ell = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} c_j \omega_N^{j\ell}. \tag{6.4}$$

Hence, $|\varphi\rangle = \text{QFT}\,|\psi\rangle$,

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_\ell \\ \vdots \\ d_{2^n-1} \end{bmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \dots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \dots & \omega_N^{2(N-1)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \dots & \omega_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \\ \vdots \\ c_{2^n-1} \end{bmatrix}. \tag{6.5}$$

It can be seen from Equations 6.4 and 6.5 that the output state vector $\{d_1, d_2, \dots, d_\ell, \dots d_{2^n-1}\}$ is given by the action of the matrix $F$ on the input state vector $\{c_1, c_2, \dots, c_j, \dots c_{2^n-1}\}$.

Things seem to be quite straightforward, and indeed they have been. Now, we try to reformulate the QFT formula in a way that is convenient to implement on a quantum computer. Following this, we will design a circuit that implements the desired operations. Owing to its linear nature, it is sufficient to develop a circuit that correctly implements the QFT for the basis states $\{|j\rangle\}$. [1]

- We start by considering the binary representation of the basis states $|\ell\rangle = |\ell_1\ell_2 \dots \ell_k \dots \ell_n\rangle$ with $\ell_k \in \{0,1\}\ \forall\ k = 1, 2, \dots n$.
- Then, we remind ourselves that the integer $\ell = \ell_1 2^{n-1} + \ell_2 2^{n-2} + \dots + \ell_n 2^{n-n} = \sum_{k=1}^{n} \ell_k 2^{n-k}$.
- Finally, $\frac{\ell}{2^n} = \frac{1}{2^n}\sum_{k=1}^{n} \ell_k 2^{n-k} = \sum_{k=1}^{n} \ell_k 2^{-k}$.

Using this, Equation 6.1 can be rewritten as

$$\begin{aligned} \text{QFT}\,|j\rangle &= \frac{1}{\sqrt{2^n}} \sum_{\ell_1=0}^{1}\sum_{\ell_2=0}^{1}\cdots\sum_{\ell_k=0}^{1}\cdots\sum_{\ell_n=0}^{1} \left( \underbrace{\exp\left[(i2\pi j)\sum_{l=1}^{n} k_l 2^{-l}\right] |\ell_1\ell_2 \dots \ell_k \dots \ell_n\rangle} \right) \\ &= \frac{1}{\sqrt{2^n}} \sum_{\ell_1=0}^{1}\sum_{\ell_2=0}^{1}\cdots\sum_{\ell_k=0}^{1}\cdots\sum_{\ell_n=0}^{1} \left( \overbrace{\bigotimes_{k=1}^{n} \exp\left[i2\pi j \ell_k 2^{-k}\right] |\ell_k\rangle} \right) \\ &= \frac{1}{\sqrt{2^n}} \bigotimes_{k=1}^{n} \left( \sum_{\ell_k=0}^{1} \exp\left[i2\pi j \ell_k 2^{-k}\right] |\ell_k\rangle \right), \end{aligned} \tag{6.6}$$

where, in the last step, we use the result of the following exercise.

<u>*Exercise:*</u> Show that the *"Sum of all the $n$-qubit tensor product basis states"* is the same as the *"Tensor product of $n$ sums of single qubit basis states"*.

$$\sum_{k=0}^{2^n-1} |\ell\rangle = \sum_{\ell_1=0}^{1}\sum_{\ell_2=0}^{1}\cdots\sum_{\ell_k=0}^{1}\cdots\sum_{\ell_n=0}^{1} \left( \bigotimes_{k=1}^{n} |\ell_k\rangle \right) = \bigotimes_{k=1}^{n} \left( \sum_{\ell_k=0}^{1} |\ell_k\rangle \right). \tag{6.7}$$

<u>*Hint:*</u> We can notice a pattern by starting with $n = 1, 2$.

[1] The following (cumbersome) steps might not seem very straightforward, so please do not hesitate to review them multiple times or discuss them with others.

Continuing from Equation 6.6, we get,

$$\mathrm{QFT}\,|j\rangle = \left(|0\rangle + \frac{\exp\left[i2\pi\left(\frac{j}{2}\right)\right]|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{\exp\left[i2\pi\left(\frac{j}{2}\right)\right]|1\rangle}{\sqrt{2^2}}\right) \otimes \cdots \otimes \left(\frac{\exp\left[i2\pi\left(\frac{j}{2^n}\right)\right]|1\rangle}{\sqrt{2}}\right) \tag{6.8}$$

$$\boxed{\mathrm{QFT}\,|j\rangle = \frac{1}{\sqrt{2^n}}\bigotimes_{k=1}^{n}\left(|0\rangle + \exp\left[i2\pi\left(\frac{j}{2^k}\right)\right]|1\rangle\right).} \tag{6.9}$$

### 6.1.1 The binarized decimal notation

The binarized decimal notation $(\underset{b}{.})$ divides a binary bit-string into two parts. On the left side, we have coefficients (0 or 1) for positive, increasing powers of 2, starting from 0, and on the right side, we have coefficients for negative, increasing (in magnitude) powers of 2 starting from -1.

For example,

$$ab(\underset{b}{.})cd = a2^1 + b2^0 + c2^{-1} + d2^{-2}. \tag{6.10}$$

Consider $x \in \{0,1\}$, then,

$$\begin{aligned} \frac{x}{2} &= 0(\underset{b}{.})x \\ \frac{x}{2^2} &= 0(\underset{b}{.})0x \\ &\vdots \\ \frac{x}{2^k} &= 0(\underset{b}{.})\overbrace{0\ldots0}^{k-1 \text{ zeros}}x. \end{aligned} \tag{6.11}$$

For two integers $k_1 < k_2$ and $x_1, x_2 \in \{0,1\}$, we get,

$$\frac{x_1}{2^{k_1}} + \frac{x_2}{2^{k_2}} = 0(\underset{b}{.})\underbrace{\overbrace{0\ldots0}^{k_1-1 \text{ zeros}}\ x_1 0\ldots0}_{k_2-1 \text{ zeros}}x_2. \tag{6.12}$$

Now, applying this to the binary bit-string representation of an integer $j = j_1 j_2 \ldots j_n$, we get,

$$\begin{aligned} \frac{j}{2} &= \frac{j_1 j_2 \cdots j_{n-1} j_n}{2} = j_1 j_2 \ldots j_{n-1}(\underset{b}{.})j_n \\ \frac{j}{2^2} &= \frac{j_1 j_2 \cdots j_{n-2} j_{n-1} j_n}{2^2} = j_1 j_2 \ldots j_{n-2}(\underset{b}{.})j_{n-1} j_n \\ &\vdots \\ \frac{j}{2^k} &= \frac{j_1 j_2 \cdots j_{n-k} j_{n-k+1} \cdots j_n}{2^k} = j_1 j_2 \ldots j_{n-k}(\underset{b}{.})j_{n-k+1} \cdots j_n. \end{aligned} \tag{6.13}$$

Basically, whenever we divide by 2, we shift the binary decimal point to the left by one place. Hence, if we multiply a number by 2, we shift the binary decimal point to the right by one place.

*Exercise:* Show that $\frac{10011(\underset{b}{.})01}{2} = 1001(\underset{b}{.})101$ and $2 \times 10011(\underset{b}{.})01 = 10010110(\underset{b}{.})1$.

*Hint:* Try finding the decimal values for the bitstrings on the LHS and RHS for both cases. The bits to the left of the binary decimal place represent the coefficients of the non-negative powers of 2 and hence make an integer, whereas the bits to its right represent the coefficients for the negative powers of two.

We now make one final observation before concluding this cumbersome algebraic process.

$$\begin{aligned}\exp\left[i2\pi\left(\frac{j}{2^k}\right)\right] &= \exp\left[i2\pi\left(j_1j_2\dots j_{n-k}(\underset{b}{.})j_{n-k+1}\dots j_n\right)\right] \\ &= \underbrace{\exp\left[i2\pi\overbrace{(j_1j_2\dots j_{n-k})}^{\text{some integer}}\right]}_{=1}\exp\left[i2\pi\left(0(\underset{b}{.})j_{n-k+1}\dots j_n\right)\right],\end{aligned} \tag{6.14}$$

hence,

$$\exp\left[i2\pi\left(\frac{j}{2^k}\right)\right] = \exp\left[i2\pi\left(0(\underset{b}{.})j_{n-k+1}\dots j_n\right)\right]. \tag{6.15}$$

Finally, using the above results, we can rewrite Equation 6.9 as

$$\boxed{\text{QFT}\,|j\rangle = \bigotimes_{k=1}^{n}\left(\frac{|0\rangle + \exp\left[i2\pi(0(\underset{b}{.})j_{n-k+1}\dots j_n)\right]|1\rangle}{\sqrt{2}}\right).} \tag{6.16}$$

Before moving towards designing the circuit, we make a very important observation.

*The QFT maps basis states to product states and hence does not introduce any entanglement.*

### 6.1.2 The QFT circuit

To begin designing the circuit, we first try to understand what state we want on each qubit. We can see this in Equation 6.16, but we can start by simplifying our task. The immediate statement that we are about to make might not seem well-motivated at first but it shall be clearer later. If we consider an operation that swaps the states on all the qubits, that is, it swaps the state of the first qubit with the last, the state of the second qubit with the second-to-last and so on. We denote this operation by $\text{SWAP}_n$ and $\text{SWAP}_n\,|q_1q_2\dots q_{n-1}q_n\rangle = |q_nq_{n-1}\dots q_2q_1\rangle$ and $\text{SWAP}_n\text{SWAP}_n = \mathbb{1}_n$. Concisely, it swaps the state of the qubit $k$ with that of qubit $n-k+1$. If we apply this operation to the QFT in Equation 6.16, we get the swapped-QFT,

$$\text{SWAP}_n\,\text{QFT}\,|j\rangle = \bigotimes_{k=1}^{n}\left(\frac{|0\rangle + \exp\left[i2\pi(0(\underset{b}{.})j_k\dots j_n)\right]|1\rangle}{\sqrt{2}}\right). \tag{6.17}$$

Now, the relative phases (the exponents in front of $|1\rangle$) seem to be much more systematic. That is, on the $k^{\text{th}}$ qubit, we want the relative phase to be proportional to $0(\underset{b}{.})j_k\dots j_n$.

We will now try to develop an intuition about the circuit that will help us achieve this swapped-QFT operation. In fact, to keep things simpler, we will just try to make a circuit that has $n$ qubits and when given the input $|j\rangle = |j_1 j_2 \dots j_n\rangle$, prepares a state on the first qubit $|\psi_{q_1}\rangle$ which is identical to the state we need on the first qubit in Equation 6.16. That is,

$$|\psi_{q_1}\rangle = \frac{|0\rangle + \exp\left[i2\pi(0(\underset{b}{.})j_1 j_2 \dots j_n)\right]|1\rangle}{\sqrt{2}}. \tag{6.18}$$

At first glance, the process of preparing this state does not seem obvious, so let's start with small steps. Since we want to introduce relative phases, we will begin by looking at the simplest way of introducing relative phases–rotation unitaries. Consider the action of the rotation unitary $U_\theta$ on the states $|0\rangle, |1\rangle$ and $|+\rangle$.

$$\begin{aligned} U_\theta &= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}. \\ U_\theta |0\rangle &= |0\rangle \\ U_\theta |1\rangle &= e^{i\theta}|1\rangle \\ U_\theta |+\rangle &= \frac{U_\theta|0\rangle + U_\theta|1\rangle}{\sqrt{2}} = \frac{|0\rangle + e^{i\theta}|1\rangle}{\sqrt{2}}. \end{aligned} \tag{6.19}$$

Hence, we have successfully introduced a relative phase $e^{i\theta}$ in the state $|+\rangle$. However, in our required state $|\psi_{q_1}\rangle$, we have a phase that depends on the states of other quits $|j_2\rangle, |j_3\rangle, \dots |j_n\rangle$. The most straightforward way to introduce dependence on the state of another qubit is to use a controlled gate. So let's try to see how the controlled version of this unitary C-$U_\theta = |0\rangle\langle 0| \otimes \mathbb{1} + |0\rangle\langle 0| \otimes U_\theta$ acts on the state $|c+\rangle$, where $|c\rangle$ is the state of the control qubit.

$$\begin{aligned} \text{C-}U_\theta |0+\rangle &= |0\rangle \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ \text{C-}U_\theta |1+\rangle &= |1\rangle \otimes \frac{|0\rangle + e^{i\theta}|1\rangle}{\sqrt{2}} \end{aligned} \tag{6.20}$$

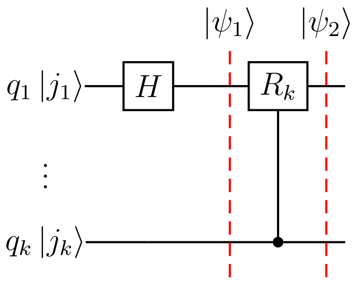
Concisely, we can write this as

$$\boxed{\text{C-}U_\theta |c+\rangle = |c\rangle \otimes \frac{|0\rangle + e^{i\theta c}|1\rangle}{\sqrt{2}}}. \tag{6.21}$$

Now, for designing the circuit, we define the unitary

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{i2\pi/2^k} \end{bmatrix}, \tag{6.22}$$

and now consider the following controlled circuit.



(6.23)

*Exercise:* Show that the action of Hadamard on a single qubit basis state gives

$$H\left|x\right\rangle = \frac{\left|0\right\rangle + \exp[\left(i\frac{2\pi}{2}x\right)]}{\sqrt{2}} \;\forall\; x = 0, 1. \tag{6.24}$$

*Hint:* The simplest way would be to write the action of the Hadamard gate on the two basis states and show it matches the required form above.

The states are

$$\begin{aligned}\left|\psi_1\right\rangle &= (H\left|j_1\right\rangle) \otimes \left|j_k\right\rangle = \left(\frac{\left|0\right\rangle + \exp\left[i2\pi\left(\frac{j_1}{2}\right)\right]\left|1\right\rangle}{\sqrt{2}}\right) \otimes \left|j_k\right\rangle \\ \left|\psi_2\right\rangle &= \frac{\left|0\right\rangle + \exp\left[i2\pi\left(\frac{j_1}{2}\right)\right]\exp\left[i2\pi\left(\frac{j_k}{2^k}\right)\right]\left|1\right\rangle}{\sqrt{2}} \otimes \left|j_k\right\rangle\end{aligned} \tag{6.25}$$

where the $j_k$ in the exponent indicates that the unitary acts only when the control qubit is in state $\left|1\right\rangle$, that is, $j_k = 1$. Using the binarized decimal notation, we know, $\frac{j_1}{2} = 0(\underset{b}{.})j_1$ and $\frac{j_k}{2^k} = 0(\underset{b}{.})\underbrace{0\ldots0}_{k-1 \text{ zeros}}\, j_k$. Hence, we get,

$$\begin{aligned}\left|\psi_2\right\rangle &= \frac{\left|0\right\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})j_1 + 0(\underset{b}{.})\overbrace{0\ldots0}^{k-1 \text{ zeros}}\, j_k\right)\right]\left|1\right\rangle}{\sqrt{2}} \otimes \left|j_k\right\rangle . \\ &= \frac{\left|0\right\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})j_1\,\overbrace{0\ldots0}^{k-2 \text{ zeros}}\, j_k\right)\right]\left|1\right\rangle}{\sqrt{2}} \otimes \left|j_k\right\rangle\end{aligned} \tag{6.26}$$

Now, instead of directly applying a controlled unitary from $q_k$, if we serially apply (one after another) such controlled unitaries controlled by all qubits ($q_2$ to $q_n$) on the first qubit, as shown in Figure 6.1, we get,

$$\left|\Psi_1\right\rangle = \underbrace{\frac{\left|0\right\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})j_1j_2\ldots j_n\right)\right]\left|1\right\rangle}{\sqrt{2}}}_{\text{State of the }1^{\text{st}}\text{ qubit }\left|\psi_{q_1}\right\rangle} \otimes \left|j_2\ldots j_n\right\rangle . \tag{6.27}$$

We note that the state of the other qubits has not been altered at all. Now, we apply a similar sequence of gates on qubit $q_2$, starting with the Hadamard. However, now instead of $q_k$ controlling $R_k$ (as in the case of $q_1$), here, in the case of $q_2$, the qubit $q_k$ will control the gate $R_{k-1}$. That is, an $R_2$ controlled by $q_3$, an $R_3$ controlled by $q_4$ and so on until an $R_{n-1}$ controlled by $q_n$, as shown in Figure 6.2.

[2] We only denote the first wire dotted since there are hidden operations happening on the first qubit. However, we do not need to do so for the other qubits, as there are no operations happening on them.

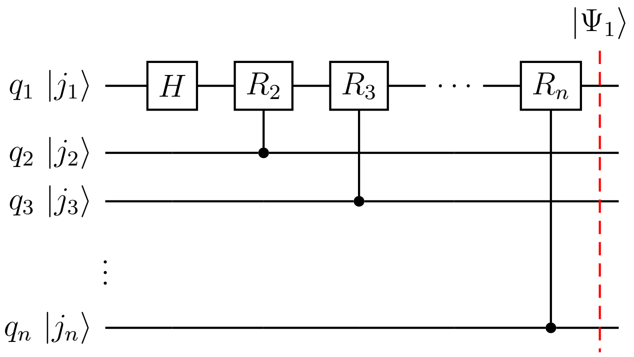


Figure 6.1: Serially applying controlled unitaries to the first qubit.[2]

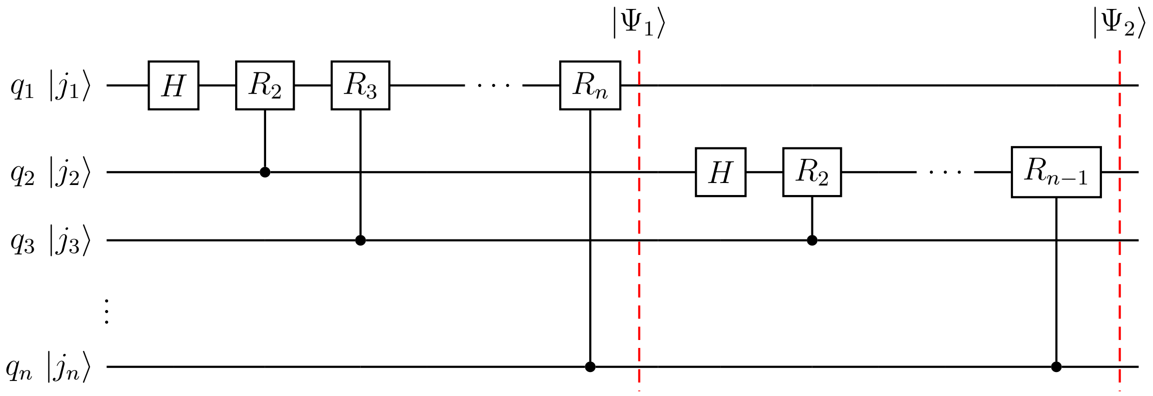


Figure 6.2: Serially applying controlled unitaries to the second qubit.

After these two sets of serial unitary applications, the total state of the system becomes

$$|\Psi_2\rangle = \underbrace{\frac{|0\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})j_1j_2\ldots j_n\right)\right]|1\rangle}{\sqrt{2}}}_{|\psi_{q_1}\rangle} \otimes \underbrace{\frac{|0\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})j_2\ldots j_n\right)\right]|1\rangle}{\sqrt{2}}}_{|\psi_{q_2}\rangle} \otimes |j_3\ldots j_n\rangle\,. \tag{6.28}$$

Continuing this process, the operations on the $k^{\text{th}}$ qubit will be

$$q_k\ \ |j_k\rangle - \boxed{H} - \boxed{\mathrm{C}_{q_{k+1}}\text{-}R_2} - \cdots - \boxed{\mathrm{C}_{q_n}\text{-}R_{n-k+1}} - |\psi_{q_k}\rangle \tag{6.29}$$

where, $\mathrm{C}_{q_x}$-$R_y$ means applying the gate $R_y$ controlled by the qubit $q_x$. In fact, the Hadamard gate on $q_k$ can be thought of as $\mathrm{C}_{q_k}$-$R_1$, and we get the sequence of gates

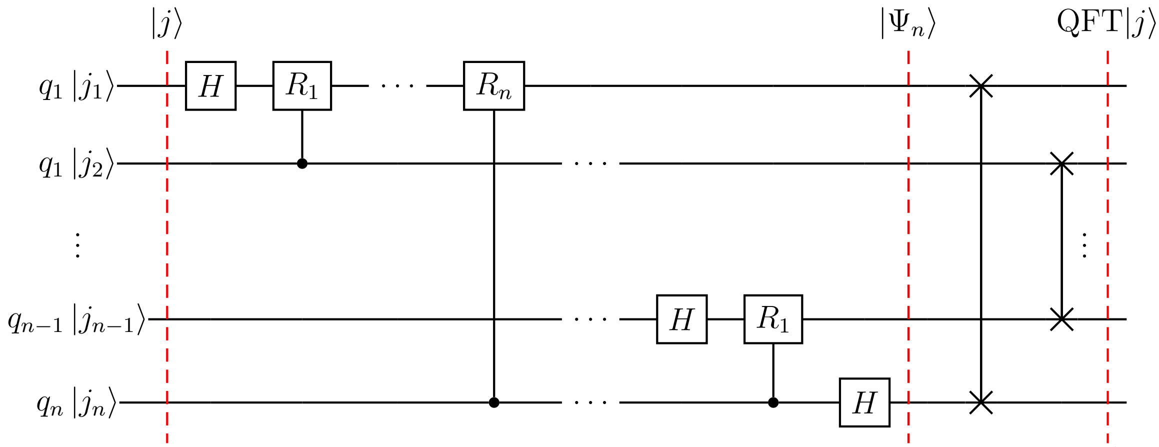
$$q_k\ \ |j_k\rangle - \boxed{\mathrm{C}_{q_k}\text{-}R_1} - \cdots - \boxed{\mathrm{C}_{q_x}\text{-}R_{x-k+1}} - \cdots - \boxed{\mathrm{C}_{q_n}\text{-}R_{n-k+1}} - |\psi_{q_k}\rangle \tag{6.30}$$

with $k \leq x \leq n$. The state of the $k^{\text{th}}$ qubit will be

$$|\psi_{q_k}\rangle = \frac{|0\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})j_kj_{k+1}\ldots j_n\right)\right]|1\rangle}{\sqrt{2}} \tag{6.31}$$

and the total state will be

$$|\Psi_k\rangle = \bigotimes_{k'=1}^{k} |\psi_{q_{k'}}\rangle\,. \tag{6.32}$$



(6.34)

Figure 6.3: The complete QFT circuit.

Continuing this, when we reach the $n^{\text{th}}$ qubit, we apply only a single Hadamard gate since there are no further qubits that will control the $R_x$ operations. Finally, after applying these cascaded stages on all $n$ qubits, the state will be

$$|\Psi_n\rangle = \bigotimes_{k=1}^{n} |\psi_{q_k}\rangle = \bigotimes_{k=1}^{n} \left( \frac{|0\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})j_k j_{k+1}\dots j_n\right)\right]|1\rangle}{\sqrt{2}} \right). \tag{6.33}$$

Which is the result of the QFT applied on $|j\rangle$, but with a reversed qubit ordering. We can simply verify this claim by replacing $k$ with $n-k$ in Equation 6.33. This replacement of index essentially means swapping all the qubits, that is, exchanging the first qubit with the last, the second qubit with the second-to-last, and so on. We denote this operation by $\mathrm{SWAP}_n$ and $\mathrm{SWAP}_n\,|q_1q_2\dots q_{n-1}q_n\rangle = |q_nq_{n-1}\dots q_2q_1\rangle$. It swaps the states of the qubit $k$ with that of qubit $n-k+1$. Applying this to $|\Psi_n\rangle$, we get,

$$\begin{aligned} \mathrm{SWAP}_n\,|\Psi_n\rangle &= \mathrm{SWAP}_n\left(\bigotimes_{k=1}^{n}|\psi_{q_k}\rangle\right) \\ &= \bigotimes_{k=1}^{n}|\psi_{q_{n-k+1}}\rangle \\ &= \bigotimes_{k=1}^{n}\left(\frac{|0\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})j_{n-k+1}\dots j_n\right)\right]|1\rangle}{\sqrt{2}}\right) = \mathrm{QFT}\,|j\rangle\,. \end{aligned} \tag{6.35}$$

Therfore, we get $\mathrm{QFT}\,|j\rangle = \mathrm{SWAP}_n\,|\Psi_n\rangle$. The complete QFT circuit is shown in Figure 6.3.

Having established that the proposed circuit indeed performs the QFT correctly, we try to understand the key idea behind the QFT circuit a bit better by emphasizing on the subtleties.

- First, we purposely designed a circuit that gives us the QFT but with a reversed qubit ordering. The reason for this choice is just to simplify the ordering of the controlled unitaries.
- Second, we leveraged the separable nature of the output, to one-by-one construct the required state on each qubit. However, it is important to preserve the state of the $k^{\text{th}}$ qubit $|j_k\rangle$ until we have constructed the required states on all the qubits above it since $j_k$ appears in the exponents in all the preceding states from $|\psi_{q_1}\rangle$ to $|\psi_{q_{k-1}}\rangle$.
- And finally, we swapped the qubits to get the QFT in the correct qubit ordering.

*Exercise:* Try to define the inverse QFT (IQFT) and design a circuit for the same.

*Hint:* Use the fact that $\text{IQFT} \cdot \text{QFT}\,|j\rangle = |j\rangle$ to first define the IQFT. In fact, since the QFT is unitary, the $\text{IQFT} = \text{QFT}^\dagger$. Then, for the circuit, try to undo the operations we performed systematically one by one. The identity $(AB)^\dagger = B^\dagger A^\dagger$, might help.

Now, we shall look at an application of the QFT.

## 6.2 Quantum phase estimation

The quantum phase estimation algorithm is (un)arguably the most quantum algorithm among the fault-tolerant ones, making it the most important application of the QFT. It is in fact, a central component in Shor's factoring algorithm and many other algorithms. QPE relies on the phase-kickback mechanism and the QFT, both of which, we have studied already.

Before presenting the full algorithm, we shall look at a simplified case

**Problem statement:** We are given an $n$-qubit unitary $U$ and one of its eigenvectors $|\mu\rangle$. Since unitary operators have unit norm eigenvalues, we can represent the eigenvalue associated with $|\mu\rangle$ as $e^{i2\pi\varphi}$ for some $0 \leq \varphi \leq 1$. The task is to find $\varphi$.

We may think that finding $\varphi$ is a trivial task, we just need to apply $U$ on $|\mu\rangle$ and check the phase that appears. However, it is important to note that the phase is a global phase and hence cannot be physically measured. In principle, we cannot distinguish between $|\mu\rangle$ and $e^{i2\pi\varphi}\,|\mu\rangle$. So how do we go about this task?

We begin by making an observation. Since $0 \leq \varphi \leq 1$, the binary representation of $\varphi$ can be written as $\varphi = 0(\underset{b}{.})\varphi_1\varphi_2\ldots\varphi_t$ (assuming we can represent $\varphi$ exactly using $t$ bits). Hence, to represent $\varphi$ we will require $t$-qubits. This will be our first register $T$. register is a collection of qubits that are addressed by one name–analogous to a classical register which is a set of a finite number of classical bits. Then, we will also require a second register to load the state $|\mu\rangle$ and perform operations on it to recover the phase. Now, it is natural to guess that the operations we need to perform will be somehow related to the unitary $U$, since it induces the phase we want to recover on $|\mu\rangle$.

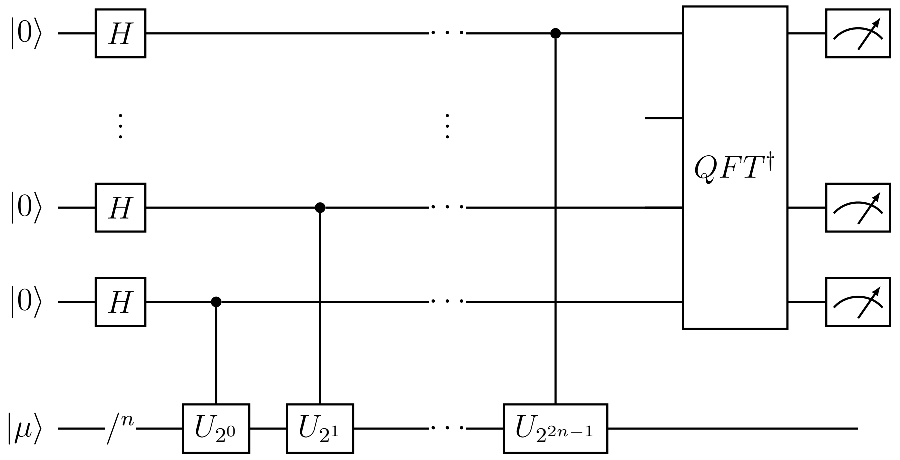
Just like the QFT encodes the binary bit-string representation of an integer into relative phases, the IQFT can recover the integer from such relative phases. For our problem, the complete integer with all the $t$-bits would be $\varphi_1\varphi_2\dots\varphi_t = 2^t\varphi = \varphi$[3]. And the QFT of this binary bitstring is

$$\mathrm{QFT}\,|\varphi\rangle = \bigotimes_{k=1}^{t}\left(\frac{|0\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})\varphi_{t-k+1}\dots\varphi_t\right)\right]}{\sqrt{2}}\right)$$

$$\mathrm{QFT}\,|\varphi\rangle = \left(\frac{|0\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})\varphi_t\right)\right]}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})\varphi_{t-1}\varphi_t\right)\right]}{\sqrt{2}}\right) \otimes \cdots$$
$$\cdots \otimes \left(\frac{|0\rangle + \exp\left[i2\pi\left(0(\underset{b}{.})\varphi_1\dots\varphi_t\right)\right]}{\sqrt{2}}\right). \tag{6.36}$$

If we prepare the above state, then we can simply perform the inverse-QFT ($\mathrm{QFT}^\dagger$) and obtain $\Phi = 2^t\varphi$. Now, in the above equation, we observe that the relative phases are simply $\varphi$ multiplied with different powers of 2. To illustrate, $2\varphi = 2[0(\underset{b}{.})\varphi_1\varphi_2\dots\varphi_t] = \varphi_1(\underset{b}{.})\varphi_2\dots\varphi_t$, and similarly $2^k\varphi = \varphi_1\dots\varphi_k(\underset{b}{.})\varphi_{k+1}\dots\varphi_t$. Since we want our first relative phase (on our first qubit) to be $0(\underset{b}{.})\varphi_t$, we shall only consider powers of 2 up to $t-1$[4].

Further, introducing a relative phase is exactly what we did while building the QFT circuit and in previous lectures (like with Deutsch-Jozsa) using the phase kickback technique. Like before, we begin by taking small steps and try to introduce the relative phase we want ($0(\underset{b}{.})\varphi_1\dots\varphi_t = \varphi$) on the $t^{\text{th}}$ bit of the register $T$.



The trick of applying $H^{\otimes t}$ to $|0\rangle$ is very common, because it gives the total superposition state:

$$|00\cdots0\rangle + |00\cdots1\rangle + |00\cdots10\rangle + |00\cdots11\rangle + \cdots + |11\cdots1\rangle \tag{6.37}$$

and everything that follows is then applied to all states of the computational basis in parallel. Since $H^{\otimes t}$ is separable, we can inspect the state of each of the $t$ qubits.

[3] We can think of this operation as simply the binary equivalent of converting a decimal $t$-digit floating point number (between 0 and 1) to an integer by multiplying it by $10^t$.

[4] Another reason is the fact that $2^t\varphi$ is an integer and hence induces a trivial phase $e^{i2\pi(2^t\varphi)} = 1$.

1st qubit:

$$|0\rangle \xrightarrow{H} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \xrightarrow{C-U^{2^{t-1}}} \frac{|0\rangle + e^{2\pi i(2^{t-1}\varphi)}\,|1\rangle}{\sqrt{2}} \tag{6.38}$$

This happens because, if we focus on the qubit 1 of the 1st register and on the 2nd register,

$$|0\rangle|u\rangle \xrightarrow{H} \frac{|0\rangle + |1\rangle}{\sqrt{2}}|u\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle|u\rangle + |1\rangle|u\rangle\right) \tag{6.39}$$

$$\xrightarrow{C\text{-}U^{2^{t-1}}} \frac{1}{\sqrt{2}}\left(|0\rangle|u\rangle + |1\rangle U^{2^{t-1}}|u\rangle\right) \tag{6.40}$$

$$= \frac{1}{\sqrt{2}}\left(|0\rangle|u\rangle + |1\rangle e^{2\pi i\varphi 2^{t-1}}|u\rangle\right) \tag{6.41}$$

$$= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(2^{t-1}\varphi)}|1\rangle\right)|u\rangle \tag{6.42}$$

The fact that the phase coming from the controlled operation appears back into the control qubit, because we used the Hadamard gate first, to obtain a full superposition, is called phase kickback, and is very commonly used in quantum algorithms.

We can now apply the same reasoning to all qubits of the 1st register. The result of the dashed line before the inverse QFT is:

$$|0\rangle + e^{2\pi i(2^{t-1}\varphi)}|1\rangle,\ |0\rangle + e^{2\pi i(2^{t-2}\varphi)}|1\rangle,\ \cdots,\ |0\rangle + e^{2\pi i(2^{0}\varphi)}|1\rangle \tag{6.43}$$

Notice that, if $\varphi = 0.\varphi_1\varphi_2\cdots\varphi_t$, then

$$2^{t-1}\varphi = 2^{t-1}\left(\frac{\varphi_1}{2} + \frac{\varphi_2}{4} + \cdots + \frac{\varphi_t}{2^t}\right) = \left(2^{t-1}\varphi_1 + 2^{t-2}\varphi_2 + \cdots + \frac{\varphi_t}{2}\right) \tag{6.44}$$

all terms in the sum are integers, except the last one. Then, all the integers disappear because they enter as $e^{2\pi ik} = 1$. Then we are left with $e^{2\pi i\varphi_t/2}$.

In the same way, the second qubit goes into:

$$\frac{\left(|0\rangle + e^{2\pi i0(\underset{b}{.})\varphi_1\varphi_2\cdots\varphi_t}|1\rangle\right)}{\sqrt{2}}. \tag{6.45}$$

and so on. The state before the inverse QFT is

$$\frac{1}{\sqrt{2^t}}\left(|0\rangle + e^{2\pi i0(\underset{b}{.})\varphi_t}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i0(\underset{b}{.})\varphi_{t-1}\varphi_t}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i0(\underset{b}{.})\varphi_1\cdots\varphi_t}|1\rangle\right). \tag{6.46}$$

This matches the state we needed. And now we see that application of the inverse QFT to register 1 gives as a final state:

$$|\varphi\rangle|u\rangle \rightarrow |\varphi\rangle|u\rangle = |\varphi_1\varphi_2\cdots\varphi_t\rangle|u\rangle \tag{6.47}$$

where $\varphi_i = 0, 1$.

The state in register 1 is a state of the computational basis. In quantum computing, we always know how to measure the qubits in the computational basis, that is, with an operator diagonal in the same basis which can read out with 100% probability if the qubit is 0 or 1.

Then by simply measuring all the qubits in the 1st register, we obtain $\varphi_1, \varphi_2, \cdots, \varphi_t$, namely the value of $\varphi$.

**Questions:**

1. What is the number of gates as a function of $n$?
2. How much must be $t$?
3. What happens if $2^t\varphi$ is not an integer number?

Questions 2 and 3 are linked to each other. We will see that if $2^t\varphi$ is not an integer, then we will obtain $\tilde{\varphi} = \lfloor 2^t\varphi \rfloor$ (the integer part of $2^t\varphi$). With high probability, the higher $t$, the longer $t$.

It is then enough to choose $t$ large enough and run the algorithm several times, to see which result for $\varphi$ one gets most frequently. One can show that if we want $\varphi$ accurate to $m$-bits, with success probability at least $1 - \epsilon$, we must choose:

$$t = m + \left\lceil \log\left(2 + \frac{1}{2\epsilon}\right) \right\rceil . \tag{6.48}$$

# Chapter 7

# Shor's Factoring Algorithm

## 7.1 Shor's Factoring Algorithm

Shor's factoring algorithm finds one factor of an integer $N$ in a time:

$$O((\log N)^3) \quad \text{with probability}\, O(1). \tag{7.1}$$

The best known classical algorithm, the general number field sieve, is sub-exponential:

$$O\left(e^{1.9(\log N)^{1/3}(\log\log N)^{2/3}}\right), \tag{7.2}$$

so the advantage is "almost" exponential.

The algorithm relies on some number-theoretic results and on the *order-finding problem*, which can be solved efficiently on a quantum computer thanks to QPE.

Here, we will take the shortest path and give all the number-theoretical results without proof.

First, let's define the order-finding problem. Let $N$ be an integer. Consider the set:

$$\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N : \gcd(a, N) = 1\} \tag{7.3}$$

where $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$.

Then, considering the multiplication modulo $N$, $\mathbb{Z}_N^*$ forms a group:

- If $a, b \in \mathbb{Z}_N^*$, then $ab \pmod N \in \mathbb{Z}_N^*$.
- If $a \in \mathbb{Z}_N^*$, then $\exists b \in \mathbb{Z}_N^*$ such that $ab \pmod N = 1$, and we call $b = a^{-1}$.
- Of course, $a \cdot 1 \pmod N = 1 \cdot a \pmod N = a$.

Remember that the *order* of a group $G$ is the number of its elements.

**Definition:** The order of an element $a \in G$ is the smallest positive integer $r$ such that:

$$a^r = e. \tag{7.4}$$

**Lagrange's Theorem:** For finite groups $G$, the order $r$ of an element of $G$ is a divisor of the order of $G$ (denoted as $\text{ord}(G)$).

**Order-finding problem:**
Given an integer $N$, and an integer $a \in \mathbb{Z}_N^*$, find the order of $a$, i.e., find the smallest positive integer $r$ such that:

$$a^r = 1 \, (\mathrm{mod}\, N). \tag{7.5}$$

## 7.2 Order Finding

$$\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N : \gcd(a, N) = 1\}. \tag{7.6}$$

$$\mathbb{Z}_N = \{0, 1, \ldots, N-1\}. \tag{7.7}$$

The number of elements in $\mathbb{Z}_N^*$ determines a function called the Euler $\varphi$-function:

$$\varphi(N) = \text{number of elements in}\, \mathbb{Z}_N^*. \tag{7.8}$$

Order finding consists in finding the smallest integer $r \neq 0$ such that:

$$a^r \,\mathrm{mod}\, N = 1. \tag{7.9}$$

This exists and is finite because of Euler's Theorem:

$$a^{\varphi(N)} \,\mathrm{mod}\, N = 1. \tag{7.10}$$

Also, Lagrange's Theorem states that $r$ for a group $G$ is a divisor of the order of $G$.

*Example:*
$a = 4$, $N = 35$:

$$4^1 = 4,\ 4^2 = 16,\ 4^3 = 29,\ 4^4 = 11,\ 4^5 = 9,\ 4^6 = 1. \tag{7.11}$$

So $r = 6$.

Link between order finding and period finding in the case of $\mathbb{Z}_N^*$. Consider the function

$$f(x) = a^x \,\mathrm{mod}\, N \quad \text{where}\, a \in \mathbb{Z}_N^*. \tag{7.12}$$

The period of $f(x)$ is the smallest integer $r \neq 0$ such that $a^r \,\mathrm{mod}\, N = 1$, because $a^0 \,\mathrm{mod}\, N = 1$. In fact, if $a$ is prime to $N$, i.e., $\gcd(a, N) = 1$, then if $a^2 < N$, $a^2$ is also in $\mathbb{Z}_N^*$, i.e., $\gcd(a^2, N) = 1$ ($a^2$ has the same distinct prime factors as $a$, and per hypothesis they do not divide $N$).

If $a^2 > N$, then we subtract $N$ from $a^2$, and apply the same considerations to the remainder. (If $a^2 = N$, then $a^2 \,\mathrm{mod}\, N = 0$, and this is a trivial case for factoring).

If, however, $a^r \,\mathrm{mod}\, N = 1$, then $a^{r+1} \,\mathrm{mod}\, N = a \,\mathrm{mod}\, N$, etc.

So $f(x)$ is periodic, and finding the period $r$ coincides to the order finding problem for the group $G = \mathbb{Z}_N^*$.

The quantum order finding is just the QPE applied to the unitary operation:

$$U|y\rangle = |xy \, (\mathrm{mod\ N})\rangle \tag{7.13}$$

with $y \in \{0, 1\}^L$ and $L = \lceil \log_2 N \rceil$ is the number of bits required to express $N$ (and assume that $U|y\rangle = |y\rangle$ if $N \leq y \leq 2^L - 1$).

In fact, we see that for $0 \leq s \leq r-1$,

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[-\frac{2\pi i s k}{r}\right] |x^k \bmod N\rangle \tag{7.14}$$

are eigenstates of $U$:

$$U|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[-\frac{2\pi i s k}{r}\right] |x^{k+1} \bmod N\rangle \tag{7.15}$$

$$= \exp\left[\frac{2\pi i s}{r}\right] |u_s\rangle, \tag{7.16}$$

where $r$ is the order of $x$ modulo $N$.

**Two problems:**
First, implement the initial state $|u_s\rangle$ (*we don't know* $r$). Second, write an efficient quantum code for the oracle $C$-$U$.

$$U|1\rangle = |x \bmod N\rangle \tag{7.17}$$

$$U|x\rangle = |x^2 \bmod N\rangle \tag{7.18}$$

$$\vdots \tag{7.19}$$

$$U|x^{r-1}\rangle = |x^r \bmod N\rangle = |1\rangle. \tag{7.20}$$

Then:

$$U\frac{|1\rangle + |x \bmod N\rangle + \cdots + |x^{r-1} \bmod N\rangle}{\sqrt{r}} = \frac{|x \bmod N\rangle + |x^2 \bmod N\rangle + \cdots + |1\rangle}{\sqrt{r}} \tag{7.21}$$

is an eigenstate of $U$.

Same for:

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i s k/r} |x^k \bmod N\rangle. \tag{7.22}$$

The second problem can be solved efficiently with a circuit with $\mathcal{O}(L^3)$ gates. The method is called modular exponentiation. We won't describe it in detail (*see Nielsen-Chuang*).

For the state preparation problem, notice that:

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle. \tag{7.23}$$

If we input $|1\rangle$, by linearity we will find one of the phases $s/r$ with probability at least $1 - \frac{\epsilon}{r}$ if we choose:

$$t = 2L + 1 + \left\lceil \log\left(2 + \frac{1}{2\epsilon}\right)\right\rceil. \tag{7.24}$$

Remember that QPE gives a $t$-bit estimate of $s$. How can we guess $s/r$ from this estimate?

We can because $s$ and $r$ are integers. There are several methods to compute the closest fraction $s/r$ to a real number $\varphi$.

The method of continued fractions is the best one. The method will give $r$ if $s/r$ is a reduced fraction (i.e., $s$ and $r$ are co-primes). If not, then there are several ways out, mostly consisting in repetitions of the algorithm to find different values of $s/r$.

All these tricks require $\mathcal{O}(\text{poly}(L))$ operations ($\mathcal{O}(L^3)$).

Shor's algorithm is a big achievement. It endangers the strong Church-Turing thesis. However, it requires fault-tolerant QC.

The largest integer factored using Shor's algorithm is 21. However, using variational algorithms, numbers as large as 291311 were factored (and adiabatic QC), using NMR QC at larger-than-room $T$.

## 7.3 Modular exponentiation

To compute $C\text{-}U^{2^j}$, we may trivially compute $C\text{-}U$ $2^j$ times. However, then the runtime would be exponential in $L$, i.e., linear in $N$. To solve this problem, there is the *modular exponentiation algorithm.*

- First, notice that there is an efficient classical algorithm for computing $a^{2^j} \bmod N$. It is enough to repeat $j$ times:

  $$a \leftarrow a^2 \bmod N. \tag{7.25}$$

  Since $j \sim L$, then this is efficient.

- Second, notice that if it is efficient classically, then it must be efficient on a quantum computer.

- Third, rewrite the sequence of $C\text{-}U^{2^j}$ as:

  $$|z\rangle|y\rangle \longrightarrow |z\rangle\, U^{z_1 2^0}\, U^{z_2 2^1} \,\cdots\, U^{z_L 2^{L-1}}|y\rangle \tag{7.26}$$

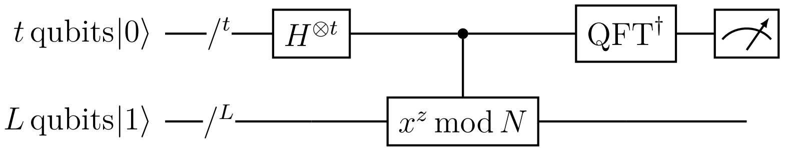
  where $z_j$ are the bits of $z$, and realize the control on the $U^{2^j}$. This gives:

  $$\longrightarrow |z\rangle|x^{z_L 2^L} \cdots x^{z_1 2^1} y \bmod N\rangle = |z\rangle|x^z y \bmod N\rangle. \tag{7.27}$$

So if we know how to compute $x^{2^j} \bmod N$, then we know how to compute:

$$x^{z_L 2^L} \bmod N = \left(x^{z_{L-1}2^{L-1}} \bmod N\right)\left(x^{z_{L-2}2^{L-2}} \bmod N\right)\cdots(x^{z_0 2^0} \bmod N). \tag{7.28}$$

Then the circuit of the order finding is:



(7.29)

## 7.4 Link Between Order Finding and Factoring

The link is made thanks to the following two theorems.

**Theorem:** $N$ is an $L$-bit integer, it is composite (i.e., not a prime). Suppose $x$ is a non-trivial solution of the equation:

$$x^2 = 1\,(\mathrm{mod}\,N), \quad 1 < x < N. \tag{7.30}$$

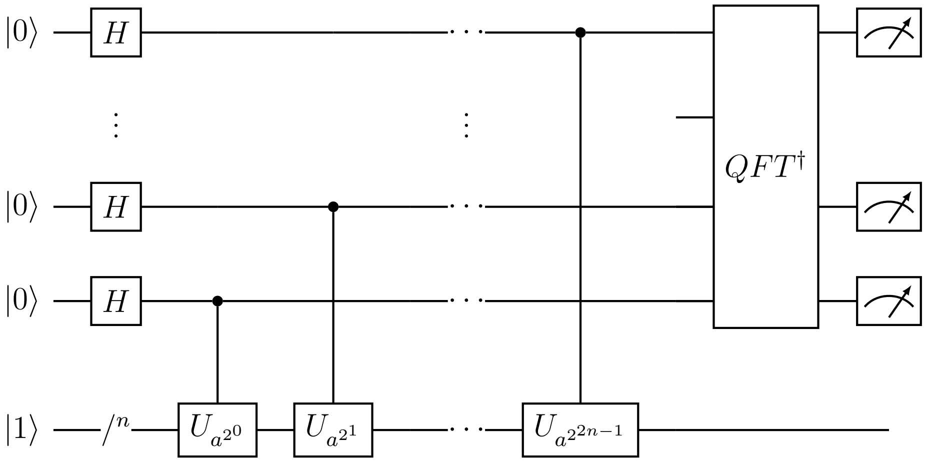
Non-trivial means that the solution is not $x = 1\,(\mathrm{mod}\,N)$ nor $x = N - 1 = -1\,(\mathrm{mod}\,N)$.

Then at least one of $\gcd(x-1, N)$ and $\gcd(x+1, N)$ is a non-trivial factor of $N$.

**Theorem:** Suppose $N = p_1^{\alpha_1} \cdots p_m^{\alpha_m}$ (the prime factorization of $N$), and $N$ is odd. Let $x$ be an integer chosen randomly (uniformly) between $1 < x \leq N - 1$, and suppose $x$ is co-prime to $N$ (i.e., $x \in \mathbb{Z}_N^*$). Let $r$ be the order of $x \,\mathrm{mod}\, N$. Then:

$$\mathrm{prob}\left[r \,\text{is even and}\, x^{r/2} \neq -1\,(\mathrm{mod}\,N)\right] \geq 1 - \frac{1}{2^m}. \tag{7.31}$$

The circuit for Shor's algorithm is give below.



## 7.5 The Algorithm is Then Simple

1. If $N$ is even, return the factor 2.
2. Determine if $N = a^b$ for integers $a \geq 3$ and $b \geq 2$. If so, return $a$.

   $$\text{For this step: if}\, N = a^b \,\text{then}\, \log_3 N = b \log_3 a \implies b = \frac{\log_3 N}{\log_3 a}. \tag{7.32}$$

   The test has $O(\log_3 N)$ complexity because one has to compute $N^{1/b} = N^{1/2},\ N^{1/3}, \ldots \log_3 N$.
3. Randomly choose $1 < x < N - 1$. If $\gcd(x, N) > 1$ (i.e., $x \notin \mathbb{Z}_N^*$), then we have found a factor, which is $\gcd(x, N)$.
4. Use the order-finding subroutine to find the order $r$ of $x \,\mathrm{mod}\, N$.
5. If $r$ is even and $x^{r/2} \neq -1\,(\mathrm{mod}\,N)$, then compute $\gcd(x^{r/2} - 1, N)$ and $\gcd(x^{r/2} + 1, N)$. If one of the two is a non-trivial factor of $N$, success. If no, fail. The success probability is $\geq \frac{3}{4}$ as $m \geq 2$. Also, if fail, one can go back to step 3 with a new randomly generated $x$ and repeat.

## 7.6 Why the Algorithm Works

The algorithm works because if $x^r = 1 \pmod{N}$ and $r$ is even, then define:

$$y = x^{r/2}. \tag{7.33}$$

If $y \neq -1 \pmod{N}$, then:

$$y^2 = mN + 1, \quad (m \, \text{an integer}), \tag{7.34}$$

$$(y+1)(y-1) = mN. \tag{7.35}$$

Thus:

$$\frac{(y+1)(y-1)}{N} \text{ is an integer.} \tag{7.36}$$

Then either $y+1$ or $y-1$ must have a common factor with $N$.

# Chapter 8

# Grover's Algorithm

## 8.1 Grover's Quantum Search Algorithm

The goal of Grover's algorithm is to carry out a search in an unstructured database *efficiently*. Take as an example *Sudoku* or the *Traveling Salesman Problem*. There are $N = 2^n$ possible configurations, and (only) one is the solution. The classical algo. consists in generating them randomly and testing if they are solutions. Both are NP-complete problems, and the best classical algorithm should then take $\mathcal{O}(N)$ attempts. Grover's algorithm takes $\mathcal{O}(\sqrt{N})$ queries to a quantum oracle that verifies a candidate solution.

This is possible because of quantum superposition. Put the initial state in:

$$|\Psi\rangle = H^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{2^n-1} |j\rangle. \tag{8.1}$$

We will see that it takes $\mathcal{O}(\sqrt{N})$ calls to the oracle to "rotate" this state close to the solution.

The oracle $\hat{O}$ is an operator defined as:

$$\hat{O}|x\rangle|q\rangle = |x\rangle|q \oplus f(x)\rangle, \tag{8.2}$$

where $x$ is an $n$-qubit register, and $q$ is a single oracle qubit such that $q = 0/1$ if $f(x) = 0/1$.

We see that the function $f(x)$ is a verifier, which tells us for each input $x$ if it is a solution to the problem by outputting $f(x) = 1$, and $f(x) = 0$ otherwise.

How to write the code for the oracle depends on the specific search problem. *Note, however, that the oracle only needs to be able to recognize the solution if given one, not to actually find it.* Example: factoring. We may factor $m$ by searching all primes $p \leq \sqrt{m}$. The oracle must only carry out a division, which can be achieved with the techniques of reversible computing.

For Grover's algo., we use the phase kickback trick. We set:

$$|q\rangle = HX|0\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{8.3}$$

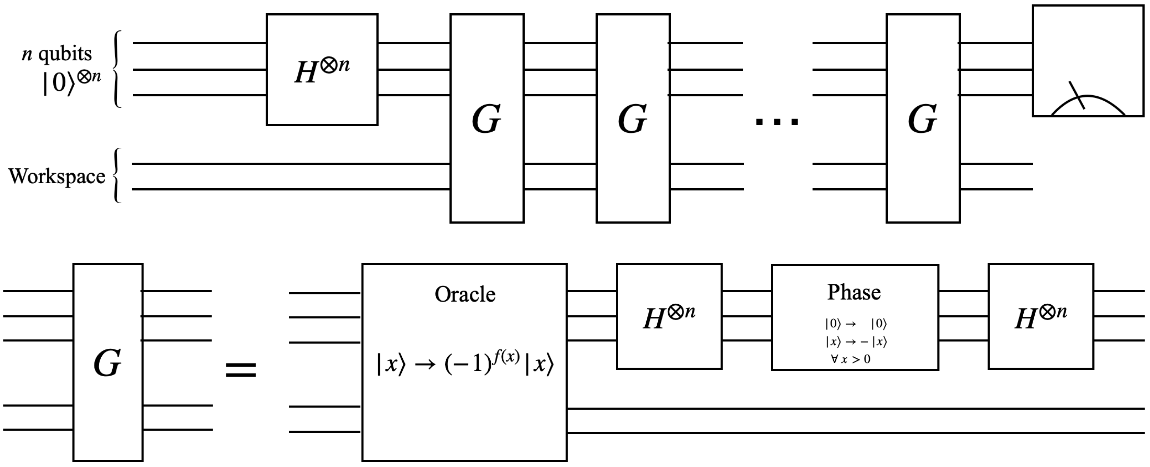


Figure 8.1: The circuit for Grover's search algorithm.

initially. Then application of the oracle gives:

$$\hat{O}|x\rangle \left( \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = (-1)^{f(x)}|x\rangle \left( \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right). \tag{8.4}$$

We see that the oracle state doesn't change. We have, however, acquired a global phase $(-1)^{f(x)}$, which will become significant when $\mathcal{O}$ is applied to a superposition of $|x\rangle$. Since the oracle state stays the same, we omit it from the discussion. Then:

$$\hat{O}|x\rangle = (-1)^{f(x)}|x\rangle. \tag{8.5}$$

We suppose that for $N$ items in the database, there are $M$ solutions, $1 \leq M \leq N$. Then one solution can be found in:

$$\mathcal{O}\left(\sqrt{\frac{N}{M}}\right) \text{ applications of } \hat{O}. \tag{8.6}$$

### 8.1.1 The Algorithm

The circuit is shown in Figure 8.1. The phase operator sends $|0\rangle$ into $|0\rangle$ and the complement subspace $|x\rangle \to -|x\rangle$. It is then expressed as $2|0\rangle\langle 0| - \mathbb{I}$. Thus:

$$G = H^{\otimes n}\left(2|0\rangle\langle 0| - \mathbb{I}\right) H^{\otimes n}\hat{O}. \tag{8.7}$$

$$G = \left(2|\psi\rangle\langle\psi| - \mathbb{I}\right)\hat{O}, \tag{8.8}$$

where:

$$|\psi\rangle = H^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle, \tag{8.9}$$

is the constant superposition state.

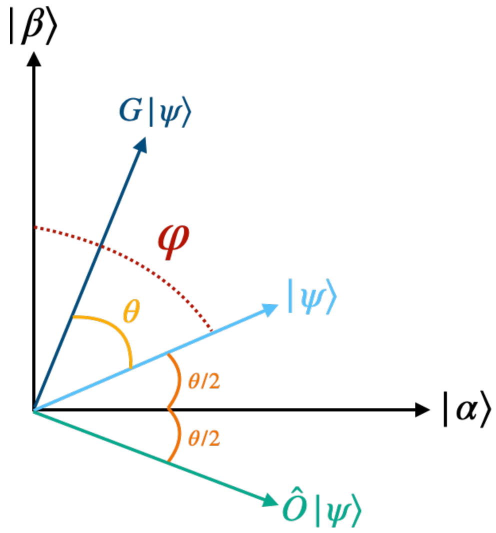


Figure 8.2: An illustration of one step of the Grover's search procedure.

### 8.1.2 Geometrical Interpretation of Grover's Algorithm

Grover's algorithm admits a nice geometrical interpretation. Define the vector $|\beta\rangle$ as:

$$|\beta\rangle = \frac{1}{\sqrt{M}} \sum_x{}' |x\rangle, \tag{8.10}$$

where $\sum_x'$ is the sum over all $x$ that are solutions, i.e., $f(x) = 1$. Define the orthogonal vector:

$$|\alpha\rangle = \frac{1}{\sqrt{N-M}} \sum_x{}'' |x\rangle, \tag{8.11}$$

where $\sum_x''$ sums over all $x$ such that $f(x) = 0$.

Then:

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle. \tag{8.12}$$

So $|\psi\rangle$ is in the plane defined by $|\alpha\rangle$ and $|\beta\rangle$. The action of the oracle $\mathcal{O}$ on a vector of this plane is:

$$\hat{O}(a|\alpha\rangle + b|\beta\rangle) = a|\alpha\rangle - b|\beta\rangle, \tag{8.13}$$

which is a reflection about the vector $|\alpha\rangle$.

The operator $2|\psi\rangle\langle\psi| - \mathbb{I}$ also performs a reflection in the same plane, about the vector $|\psi\rangle$. The result of two reflections is a *rotation*. This is best seen graphically as shown in **??**.

We see that:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|\alpha\rangle + \sin\left(\frac{\theta}{2}\right)|\beta\rangle, \tag{8.14}$$

where:

$$\cos\left(\frac{\theta}{2}\right) = \sqrt{\frac{N-M}{N}}. \tag{8.15}$$

Then:

$$G|\psi\rangle = \cos\left(\frac{3\theta}{2}\right)|\alpha\rangle + \sin\left(\frac{3\theta}{2}\right)|\beta\rangle, \tag{8.16}$$

and:

$$G^k|\psi\rangle = \cos\left(\frac{(2k+1)\theta}{2}\right)|\alpha\rangle + \sin\left(\frac{(2k+1)\theta}{2}\right)|\beta\rangle. \tag{8.17}$$

We see that by rotating by $\theta$ several times, we eventually get very close to $|\beta\rangle$. Then, a measurement on the computational basis will give $|\beta\rangle$ with high probability.

The angle $\theta$ is defined by the problem. Ideally, we need an overall rotation by an angle $\phi = \arccos\sqrt{\frac{M}{N}}$. Then we need to apply the operator $G$, $R$ times where,

$$R = \text{round}\left(\frac{\arccos\sqrt{\frac{M}{N}}}{\theta}\right). \tag{8.18}$$

### 8.1.3 Number of Applications and Probability Analysis

This brings us to an angle smaller than $\frac{\theta}{2}$ from $|\beta\rangle$. But:

$$\frac{\theta}{2} \leq \frac{\pi}{4}, \tag{8.19}$$

which implies that the measurement on the computational basis will give a solution with probability $> \frac{1}{2}$. This eventually solves the problem of finding one solution.

To better estimate the probability, we can suppose $M \ll N$. Then:

$$\theta = \sin\theta \approx 2\sqrt{\frac{M}{N}}, \tag{8.20}$$

and:

$$\frac{\theta}{2} = \sqrt{\frac{M}{N}} \ll 1, \quad P_{\text{error}} < \frac{M}{N}. \tag{8.21}$$

To estimate the number of applications $R$ of $G$, note that:

$$R \leq \frac{\pi}{2\theta}, \quad \text{since } \arccos(x) \leq \frac{\pi}{2}. \tag{8.22}$$

So since:

$$\frac{\theta}{2} \approx \sin\left(\frac{\theta}{2}\right) = \sqrt{\frac{M}{N}}, \tag{8.23}$$

$$R \leq \frac{\pi}{4}\sqrt{\frac{N}{M}} = \mathcal{O}\left(\sqrt{\frac{N}{M}}\right). \tag{8.24}$$

What if $M$ is large? If $M \approx \frac{N}{2}$, instead of running Grover, pick a value of $x$ at random, and the probability of finding a solution is $\approx \frac{1}{2}$.

If we don't know whether $M \geq \frac{N}{2}$, we can add one qubit to the database, so doubling its size, and rewrite $\hat{O}$ so that all extra entries have $f(x) = 0$. Now we have $2N$ elements and necessarily $M < \frac{2N}{2}$.

What if we don't know $M$? How do we determine $\hat{O}$?

There is an algorithm for counting the number of solutions $M$ to a search problem. It is the *quantum counting algorithm.* It is based simply on quantum phase estimation, and it also runs in $\mathcal{O}(\sqrt{N})$.

The true difficulty of Grover's algorithm is to *write an efficient oracle for the problem under analysis.*

# Chapter 9

# Digital Quantum Simulation

The original task for quantum computing as proposed by Feynman, has always been to simulate the dynamics of a quantum-mechanical system.

## 9.1 Time-Evolution Operator with Discretized Time Steps

The task of digital quantum simulation is to simulate the dynamics of an $n$-spin-$\frac{1}{2}$ system governed by the Hamiltonian $\hat{H}$ using a quantum computer.

$$\hat{H} = \sum_{j=1}^{L} \hat{h}_j = \sum_{j=1}^{L} \hat{h}_j P_j, \tag{9.1}$$

where each $\hat{h}_j = c_j P_j$ and $P_j$ are weight-$k$ Pauli operators on $n$ qubits. The weight is simply the number of qubits on which the Pauli word acts non trivially, or the number of non-identity Pauli matrices in the word. We aim to simulate the time-evolution operator $U(t) = e^{-i\hat{H}t}$.

Now, we rewrite the time-evolution operator over a time $t$ in terms of $N$ smaller steps of size $\Delta t = \frac{t}{N}$. This gives:

$$U(t) = e^{-i\hat{H}t} = \left(e^{-i\hat{H}\Delta t}\right)^N . \tag{9.2}$$

This formula is exact, as we are simply dividing the evolution into $N$ sequential applications of the same operator over shorter time intervals $\Delta t$, without any approximation yet. For each smaller time step $\Delta t$, we can now attempt to find $e^{-i\hat{H}\Delta t}$. It might seem easy at first sight but, as we will see briefly, this is not an easy task, especially when we have the Hamiltonian $\hat{H}$ made up of several non-commuting terms.

## 9.2 Zassenhaus Formula

Since $\hat{H}$ is expressed as a sum of terms $\hat{H} = \sum_{j=1}^{L} \hat{h}_j$, and in general the terms $\hat{h}_j$ do not commute, directly computing $e^{-i\hat{H}\Delta t}$ is challenging. To proceed, we apply the Zassenhaus formula, which allows us to approximate the exponential of a sum of operators as a product of exponentials:

$$e^{A+B} = e^A e^B e^{-\frac{1}{2}[A,B]} e^{\frac{1}{6}([A,[A,B]]+[B,[B,A]])} \dots \tag{9.3}$$

for two operators $A$ and $B$. In the case of our Hamiltonian, with multiple terms $\hat{h}_j$, the formula extends to:

$$e^{\sum_{j=1}^{L} \hat{h}_j} \approx \prod_{j=1}^{L} e^{\hat{h}_j} \times \text{correction terms.} \tag{9.4}$$

Each correction term involves nested commutators of the operators $\hat{h}_j$, and these terms contribute to the error of the approximation. For the purpose of digital quantum simulation, we typically use the lowest-order approximation, which keeps only the first term in the expansion and neglects the higher-order commutator terms.

## 9.3 Suzuki-Trotter Decomposition

Applying the first-order approximation of the Zassenhaus formula to $e^{-i\hat{H}\Delta t}$ with $H = \sum_{j=1}^{L} \hat{h}_j$, we obtain,

$$e^{-i\hat{H}\Delta t} \approx \prod_{j=1}^{L} e^{-i\hat{h}_j \Delta t} + O(\Delta t^2). \tag{9.5}$$

This approximation introduces an error in each time step of size $O(\Delta t^2)$, due to the neglected commutator terms.

When we apply this approximation over $N$ steps to reach a total time $t = N\Delta t$, the cumulative error over all steps scales as:

$$O(N\Delta t^2) = O(t\Delta t). \tag{9.6}$$

Thus, for a total evolution time $t$, the overall error in the first-order Suzuki-Trotter decomposition is $O(t\Delta t)$. This error decreases linearly with $\Delta t$, so improving the accuracy requires increasing $N$ (i.e., reducing $\Delta t$), leading to a finer decomposition.

Combining the above steps, we approximate the time-evolution operator $U(t) = e^{-i\hat{H}t}$ as

$$\boxed{U(t) \approx \left(\prod_{j=1}^{L} e^{-i\hat{h}_j \Delta t}\right)^N = \left(e^{-i\hat{h}_1 \Delta t} e^{-i\hat{h}_2 \Delta t} \cdots e^{-i\hat{h}_L \Delta t}\right)^N.} \tag{9.7}$$

To improve the accuracy, we can use a second-order Suzuki-Trotter expansion, which reduces the error scaling from $O(t\Delta t)$ to $O(t\Delta t^2)$, hence introducing a quadratic error decay with increasing $N$. This decomposition introduces a symmetric ordering that cancels out some of the lower-order error terms. Specifically, the second-order decomposition for a single time step is given by:

$$e^{-i\hat{H}\Delta t} \approx \prod_{j=1}^{L} e^{-i\hat{h}_j \Delta t/2} \prod_{j=L}^{1} e^{-i\hat{h}_j \Delta t/2}. \tag{9.8}$$

This means we apply each $e^{-i\hat{h}_j\Delta t/2}$ operator twice per time step, once in the forward ordering and once in reverse ordering, creating a palindromic sequence of operations. This symmetry helps to reduce the accumulation of errors associated with the non-commutativity of the terms $\hat{h}_j$.

To simulate the evolution over a total time $t = N\Delta t$, we repeat the second-order decomposition $N$ times:

$$U(t) \approx \left( \prod_{j=1}^{L} e^{-i\hat{h}_j\Delta t/2} \prod_{j=L}^{1} e^{-i\hat{h}_j\Delta t/2} \right)^N . \tag{9.9}$$

For each time step $\Delta t$, the error in the second-order expansion scales as $O(\Delta t^3)$. When applied over $N$ steps to reach the total evolution time $t = N\Delta t$, the cumulative error becomes

$$O(N\Delta t^3) = O(t\Delta t^2). \tag{9.10}$$

## 9.4 Quantum Circuit Implementation

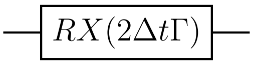
Now, we will consider the Transverse Field Ising model (TFIM) and demonstrate how to implement these Trotter steps on a quantum computer. The Hamiltonian TFIM is given by

$$\hat{H} = J \sum_{\langle i,j \rangle} \hat{Z}_i \hat{Z}_j - \Gamma \sum_i \hat{X}_i, \tag{9.11}$$

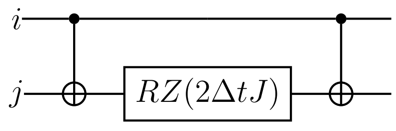
where $J$ represents the interaction strength between neighboring spins, and $\Gamma$ is the transverse field strength. Since the $Z$ operators commute among themselves but not with the $X$ operators, we decompose the Hamiltonian into two non-commuting terms.

$$\hat{H}_Z = J \sum_{\langle i,j \rangle} \hat{Z}_i \hat{Z}_j, \quad \hat{H}_X = -\Gamma \sum_i \hat{X}_i. \tag{9.12}$$

Now, the time evolution under the transverse field term $e^{i\Delta t\Gamma X}$ can be implemented as a rotation around the $x$ axis using an $RX(\theta)$ gate with an angle $\theta = 2\Delta t\Gamma$[1]. Further, if the hardware does not support $RX$ gates natively, we could also implement $RX(\theta) = HRZ(\theta)H$.



Further, the evolution under the coupling term requires a 2-qubit gate. To implement $e^{-i\Delta tJZ_iZ_j}$, we need to rotate by am angle $-\Delta tJ$ if the two spins are same and by $+\Delta tJ$ if the two spins are different. This can be realized as follows.



Similar circuits can be defined for different Hamiltonians and hardware capabilities.

[1] The input angle to the $RX$ gate is double the required angle of rotation owing to the defintion of the $RX$ gate.

# Chapter 10

# The Density Operator Formalism

## 10.1 The density operator formalism

In quantum mechanics, we often have to describe open quantum systems, i.e. systems that interact with an environment. We want to describe the system, under the influence of the environment, without having to describe the details of the environment. This is similar to what we do in thermodynamics or statistical physics, where instead of describing each single trajectory of the many molecules of a gas, we give up most of this information and keep only relevant information in terms of effective equations and average quantities like pressure, volume, temperature, etc. In quantum mechanics, giving up information about the environment is a more subtle task, as system and environment generally exist in an entangled state, as a consequence of interactions. The *density operator*, or *density matrix* formalism enables such an effective description of an open quantum system.

Consider the simplest possible representation of system and environment as two quantum bits, describing respectively the system and the environment. Assume they are in the entangled state

$$|\psi\rangle = \frac{|0_S 0_E\rangle + |1_S 1_E\rangle}{\sqrt{2}} \tag{10.1}$$

We know that for this state it does not make sense to ask the question: what is the state of the system $S$? However, if we measure an observable $\hat{O}_S$ of the system, this amounts to measuring $\hat{O} = \hat{O}_S \otimes \mathbb{1}_E$ on the whole state $|\psi\rangle$. Then the expectation value is:

$$\begin{aligned} \langle\hat{O}_S\rangle &= \langle\hat{O}\rangle = \langle\psi|\,\hat{O}\,|\psi\rangle && (10.2)\\ &= \frac{1}{2}\langle 0_S|\,\hat{O}_S\,|0_S\rangle + \frac{1}{2}\langle 1_S|\,\hat{O}_S\,|1_S\rangle && (10.3)\end{aligned}$$

It is as if $\langle\hat{O}_S\rangle$ is the result of a statistical average over an ensemble of quantum states. More precisely $\langle\hat{O}\rangle$ is the average of its value for the system in the state $|0_S\rangle$ and its value for the system in the state $|1_S\rangle$, with equal probabilities $p = 1/2$.

This notion of *statistical mixture* of states can be made formal. Consider two subsystems $S_1$ and $S_2$ with the Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$. Then the total system S is described by the states in the space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$. Suppose $\{|\phi_1\rangle, ...\}$ be the basis of $\mathcal{H}_1$ and $\{|\eta_1\rangle, ...\}$

of $\mathcal{H}_2$. Then:

$$|\psi\rangle \in \mathcal{H} \tag{10.4}$$
$$|\psi\rangle = \sum_{jk} C_{jk} |\phi_j\rangle \otimes |\eta_k\rangle \tag{10.5}$$

Assume that $S_1$ is the system and $S_2$ the environment. Consider a system observable $\hat{O}_S : \mathcal{H}_1 \to \mathcal{H}_1$. Then, in the full Hilbert space $\mathcal{H}$, the corresponding operator is $\hat{O} = \hat{O}_S \otimes \hat{\mathbb{1}}_E$. Its expectation value is

$$\begin{aligned}
\langle \hat{O}_S \rangle &= \langle\psi| \hat{O} |\psi\rangle && (10.6)\\
&= \sum_{jl}\sum_m c^*_{lm} c_{jm} \langle\phi_l| \hat{O}_S |\phi_j\rangle && (10.7)\\
&= \sum_{jl} \rho_{jl} [\hat{O}_S]_{lj} = \mathrm{Tr}(\hat{\rho}\hat{O}_S) && (10.8)
\end{aligned}$$

where $\hat{\rho}$ is an operator in $\mathcal{H}_1$ whose matrix element are:

$$\rho_{jl} = \langle\phi_j| \hat{\rho} |\phi_l\rangle = \sum_m C^*_{lm} C_{jm} \tag{10.9}$$

This is the *density matrix*, in the basis that we are considering, and $\hat{\rho}$ is the corresponding density operator. We see that for every system's observable we can compute $\langle \hat{O}_S \rangle = \mathrm{Tr}(\hat{\rho}\hat{O}_S)$.

Therefore $\hat{\rho}$ contains all info on the expectation values of observables of the system $S_1$. As we will see in a moment, it also contains all the information on the probabilities of different measurement outcomes. It is therefore a full description of the system $S_1$, when in interaction with an environment $S_2$. Let us look again at the expression of $\hat{\rho}$:

$$\begin{aligned}
\hat{\rho} &= \sum_{jl} \rho_{jl} |\phi_l\rangle \langle\phi_j| && (10.10)\\
&= \sum_{jl} |\phi_l\rangle \left( \sum_m C^*_{lm} C_{jm} \right) \langle\phi_j| && (10.11)\\
&= \sum_{jl}\sum_m |\phi_l\rangle \langle\phi_l \eta_m| |\psi\rangle \langle\psi| |\phi_j \eta_m\rangle \langle\phi_j| && (10.12)
\end{aligned}$$

Using the completeness relation $\sum_l |\phi_l\rangle \langle\phi_l| = \hat{\mathbb{1}}$, we get

$$\begin{aligned}
\hat{\rho} &= \sum_m \left( \sum_l |\phi_l\rangle \langle\phi_l| \right) \langle\eta_m| |\psi\rangle \langle\psi| |\eta_m\rangle \left( \sum_j |\phi_j\rangle \langle\phi_j| \right) && (10.13)\\
&= \sum_m \langle\eta_m| |\psi\rangle \langle\psi| |\eta_m\rangle && (10.14)\\
&= \mathrm{Tr}_E(|\psi\rangle \langle\psi|) && (10.15)
\end{aligned}$$

This is a partial trace. Notice that $\langle\eta_m| |\psi\rangle$ is not a number, and here we are actually abusing of Dirac's notation. It is a vector in $\mathcal{H}_1$ obtained by considering the full expansion

(10.5) of $|\psi\rangle$ and replacing each ket $|\eta_k\rangle$ with the inner product $\langle\eta_m|\eta_k\rangle$, i.e.,

$$\langle\eta_m|\,|\psi\rangle = \sum_{jk} C_{jk}\,|\phi_j\rangle\,\langle\eta_m|\,|\eta_k\rangle \tag{10.16}$$

$$= \sum_{jk} C_{jk}\,|\phi_j\rangle\,\delta_{mk} \tag{10.17}$$

$$= \sum_{j} C_{jm}\,|\phi_j\rangle \tag{10.18}$$

The partial trace is a very important notion for open quantum systems. It allows expressing the density operator of one subsystem starting from the density operator of the state of system and environment. The density operator $\hat{\rho}$ has three properties which can be easily proven. It is self-adjoint, it has trace one, and it is positive semidefinite, i.e.,

1. $\hat{\rho}^\dagger = \hat{\rho}$
2. $\mathrm{Tr}(\hat{\rho}) = \sum_{jm} |C_{jm}|^2 = 1$
3. $\langle\phi|\,\hat{\rho}\,|\phi\rangle \geq 0 \;\; \forall\,|\phi\rangle \in \mathcal{H}_1$

The trace formula (10.8) to compute expectation values must hold equally for an isolated system in state $|\psi\rangle$. Then, the density operator corresponding to the state $|\psi\rangle$ is given by

$$\hat{\rho} = |\psi\rangle\,\langle\psi|\;. \tag{10.19}$$

Indeed, $\langle\hat{O}\rangle = \mathrm{Tr}(\hat{\rho}\hat{O}) = \mathrm{Tr}(|\psi\rangle\,\langle\psi|\,\hat{O})$. We use the fact that the trace is independent of the choice of the basis. By choosing a basis where the first basis vector is $|\psi\rangle$ and all other basis vectors are chosen to be orthogonal to $|\psi\rangle$, we get $\langle\hat{O}\rangle = \langle\psi|\,\hat{O}\,|\psi\rangle$.

When $\hat{\rho} = |\psi\rangle\,\langle\psi|$, the system is said to be in a pure state (as opposed to a statistical mixture). A state described by $\hat{\rho}$ is pure if and only if $\mathrm{Tr}(\hat{\rho}^2) = 1$.
Proof: $\hat{\rho} = \hat{\rho}^\dagger$. Then diagonalize $\hat{\rho}$:

$$\hat{\rho} = \sum_i P_i\,|i\rangle\,\langle i| \tag{10.20}$$

where $|i\rangle$ are the eigenvectors of $\hat{\rho}$. Since $\mathrm{Tr}(\hat{\rho}) = \sum_i P_i = 1$ and $\hat{\rho}$ is positive, the relation $0 \leq P_i \leq 1$. For a system in a pure state $|\psi\rangle$, one has $P_i = 1$ for $|i\rangle = |\psi\rangle$, and $P_i = 0$ for all eigenstates orthogonal to $|\psi\rangle$. Then $\mathrm{Tr}(\hat{\rho}^2) = \sum_i P_i^2 = 1$.

In general, $\mathrm{Tr}(\hat{\rho}^2) \leq 1$. If $\mathrm{Tr}(\hat{\rho}^2) < 1$ holds strictly, then the system is in a *statistical mixture*. In this state, the properties of the system are described by the average properties of a statistical ensemble of pure states. This can be seen by computing the expectation value of a system observable $\hat{O}$ using the spectral representation of the density operator $\hat{\rho} = \sum_i P_i\,|i\rangle\,\langle i|$

$$\langle\hat{O}\rangle = \mathrm{Tr}(\hat{\rho}\hat{O}) = \sum_i \langle i|\hat{\rho}\hat{O}|i\rangle \tag{10.21}$$

$$= \sum_i P_i \langle i|\hat{O}|i\rangle \tag{10.22}$$

$$= \sum_i P_i \langle\hat{O}\rangle_i \tag{10.23}$$

So the expectation value is expressed as the statistical average of the quantum expectation value on each state $|i\rangle$ of the statistical ensemble, with the corresponding probabilities $P_i$.

We can also express the measurement probabilities with the density operator. According to the measurement postulate, if we measure $\hat{O}$ on $|\phi\rangle$ with eigenvalues $o_i$ and eigenvectors $|o_i\rangle$, then the probability of measuring the eigenvalue $o_i$ is $p(o_i) = \langle\phi|\hat{\mathbb{P}}_i|\phi\rangle$, where $\hat{\mathbb{P}}_i = |o_i\rangle\langle o_i|$ if the eigenvalue $o_i$ is non-degenerate.

We now apply the measurement postulate on the system plus environment, i.e. to the global state $|\psi\rangle$. Then

$$p(o_i) = \langle\psi|\hat{\mathbb{P}}_i \otimes \mathbb{I}|\psi\rangle \tag{10.24}$$

We note that $\hat{\mathbb{P}}_i$ is a projector i.e. $\hat{\mathbb{P}}_i^2 = \hat{\mathbb{P}}_i$ and $\hat{\mathbb{P}}_i = \hat{\mathbb{P}}_i^\dagger$. Then $\hat{\mathbb{P}}_i \otimes \mathbb{I}$ is also a projector and we can replace $\hat{\mathbb{P}}_i \otimes \mathbb{I} \to (\hat{\mathbb{P}}_i \otimes \mathbb{I})^2$. Then

$$\begin{align}
p(o_i) &= \langle\psi|\hat{\mathbb{P}}_i \otimes \mathbb{I}|\psi\rangle \tag{10.25}\\
&= \langle\psi|(\hat{\mathbb{P}}_i \otimes \mathbb{I})^2|\psi\rangle \tag{10.26}\\
&= \sum_{jk} \langle\psi|\hat{\mathbb{P}}_i \otimes \mathbb{I}|o_j\rangle|\eta_k\rangle\langle\eta_k|\langle o_j|\hat{\mathbb{P}}_i \otimes \mathbb{I}|\psi\rangle \tag{10.27}\\
&= \sum_{k} \langle\psi||o_i\eta_k\rangle\langle o_i\eta_k||\psi\rangle \tag{10.28}\\
&= \langle o_i|(\sum_k \langle\eta_k||\psi\rangle\langle\psi||\eta_k\rangle)|o_i\rangle \tag{10.29}\\
&= \langle o_i|Tr_E(|\psi\rangle\langle\psi|)|o_i\rangle \tag{10.30}\\
&= \langle o_i|\hat{\rho}|o_i\rangle \tag{10.31}
\end{align}$$

But $\hat{\rho} = \sum_i P_i|i\rangle\langle i|$. Then:

$$\begin{align}
\langle\hat{O}\rangle &= \langle o_i|\hat{\rho}|o_i\rangle \tag{10.32}\\
&= \sum_i P_i\langle o_i||i\rangle\langle i||o_i\rangle \tag{10.33}\\
&= \sum_i P_i||\hat{\mathbb{P}}_i|i\rangle||^2 \tag{10.34}\\
&= \sum_i P_i\langle\hat{O}\rangle_{|i\rangle} \tag{10.35}
\end{align}$$

which completes our interpretation of $\hat{\rho}$ as a statistical mixture. It is as if the measurement is repeated several times, each time on a state taken from a statistical ensemble of states $|i\rangle$ distributed with probability $P_i$.

If the global state of system and environment is itself a statistical mixture $\hat{\rho}$, then the above derivation can be generalized and the expectation value and measurement probabilities of a system observable $\hat{O}_S$ are expressed as

$$\begin{align}
\langle\hat{O}_S\rangle &= \mathrm{Tr}(\hat{\rho}_S\hat{O}_S) \tag{10.36}\\
p(o_i) &= \langle o_i|\hat{\rho}_S|o_i\rangle\ , \tag{10.37}
\end{align}$$

where $\hat{\rho}_S = \mathrm{Tr}_E(\hat{\rho})$. In case the eigenspace associated to the eigenvalue $o_i$ is degenerate, the expression for the expectation value remains unchanged, while the measurement probabilities are now given by

$$p(o_i) = \mathrm{Tr}(\hat{\rho}_S \hat{\mathbb{P}}_i)\,, \tag{10.38}$$

where now $\hat{\mathbb{P}}_i$ is the orthogonal projector on the degenerate eigenspace associated to the eigenvalue $o_i$.

One relevant question is how entanglement is represented in the density operator formalism. This turns out to be a quite complex question and there are no systematic ways to measure the amount of entanglement of a mixed state. However, one can generalize the notion of separable state to statistical mixtures. Assume a system $S$ composed of two subsystems $S_1$ and $S_2$. The most general separable state of $S$ is

$$\hat{\rho} = \sum_j p_j \hat{\rho}_j^{(1)} \otimes \hat{\rho}_j^{(2)}\,, \tag{10.39}$$

where $p_j$ are probabilities and $\sum_j p_j = 1$. Indeed, if $\hat{O} = \hat{O}^{(1)} \otimes \hat{O}^{(2)}$ then:

$$\begin{aligned}
\langle \hat{O} \rangle &= \mathrm{Tr}(\hat{\rho}\hat{O}) && (10.40)\\
&= \sum_j p_j \,\mathrm{Tr}(\hat{O}^{(1)} \otimes \hat{O}^{(2)}\; \hat{\rho}_j^{(1)} \otimes \hat{\rho}_j^{(2)}) && (10.41)\\
&= \sum_j p_j \,\mathrm{Tr}(\hat{O}^{(1)}\; \hat{\rho}_j^{(1)})\;\mathrm{Tr}(\hat{O}^{(2)}\; \hat{\rho}_j^{(2)}) && (10.42)\\
&= \sum_j p_j \langle \hat{O}^{(1)} \rangle \langle \hat{O}^{(2)} \rangle && (10.43)
\end{aligned}$$

which is the statistical average over the uncorrelated expectation value $\langle \hat{O}^{(1)} \otimes \hat{O}^{(2)} \rangle = \langle \hat{O}^{(1)} \rangle \langle \hat{O}^{(2)} \rangle$.

Another important property of the density matrix is convexity. Density operators live in the Liouville space $S(\mathcal{H})$. The convexity implies that if $\hat{\rho}_1 \in S(\mathcal{H})$ and $\hat{\rho}_2 \in S(\mathcal{H})$ then $\lambda\hat{\rho}_1 + (1-\lambda)\hat{\rho}_2 \in S(\mathcal{H})$ for $\lambda \in [0,1]$.

Pure states, represented as $|\psi\rangle\langle\psi|$, are not decomposable into convex combinations of density operators. Therefore, they reside on the boundaries of the space $S(\mathcal{H})$ which consists of all density matrices. A key result of this convexity is that a density matrix, representing a mixed state, can be expressed in various ways using convex linear combinations of other density matrices, highlighting the non-uniqueness of such representations.

Example: The totally mixed state, whose density operator is $\hat{\rho} = \hat{\mathbb{1}}$, once cast in matrix form it has the same matrix independently of the choice of the basis.

In conclusion, the formalism of the density operator is capable of representing all pure states and is extendable to statistical mixtures. Furthermore, it generalizes the expressions for expectation values and measurement probabilities to accommodate statistical mixtures. Consequently, within the quantum information literature, states are expressed more and more frequently as density operators, independently of their mixed or pure nature, while the state-vector formalism is declining.

### 10.1.1 Time evolution of the density operator

To fully establish the density matrix formalism, it is essential to describe the temporal evolution of a statistical mixture. The simplest scenario arises when there is no interaction with the environment over time. Consider a statistical mixture that has been prepared at time $t = 0$

$$\hat{\rho}(0) = \sum_j P_j |\phi_j\rangle \langle\phi_j| \tag{10.44}$$

. Due to the absence of coupling to the environment, each pure state $|\phi_j\rangle$ will evolve according to $|\phi_j(t)\rangle = \hat{U}(t,0)|\phi_j\rangle$, where $\hat{U}(t,0)$ is the unitary time evolution operator. Then

$$\begin{aligned}\hat{\rho}(t) &= \sum_j P_j \hat{U}(t,0) |\phi_j\rangle \langle\phi_j| \hat{U}^\dagger(t,0) & (10.45)\\ &= \hat{U}(t,0)\hat{\rho}(0)\hat{U}^\dagger(t,0)\,, & (10.46)\end{aligned}$$

The evolution thus remains unitary, and the operator $\hat{U}(t,0)$ obeys the usual equation

$$i\frac{\partial}{\partial t}\hat{U}(t,0) = \hat{H}(t)\hat{U}(t,0)\,, \tag{10.47}$$

where $\hat{U}(t,0)\hat{U}^\dagger(t,0) = \mathbb{I}$. Consequently, the equation governing the time evolution of the density operator is

$$\frac{\partial}{\partial t}\hat{\rho}(t) = -i[\hat{H}(t), \hat{\rho}(t)]\,. \tag{10.48}$$

This equation is referred to as the Von Neumann equation. It resembles the time-evolution equation for observables in the Heisenberg picture, but it has an opposite sign on the right-hand side.

What happens if the system interacts with an external environment over time? Equations $\hat{\rho}(t) = \hat{U}(t,0)\hat{\rho}(0)\hat{U}^\dagger(t,0)$ and $\dot{\hat{\rho}}(t) = -i[\hat{H}(t), \hat{\rho}(t)]$ can be generalized. The generalization of the time-evolution equation results in the celebrated Lindblad-Von Neumann Master equation, under the general assumption of a Markovian environment. In quantum computing one is almost always interested in the integral form of the time evolution. In this case, the time-evolution operator can be generalized into the notion of *quantum channel.*

To define a quantum channel, assume that the evolution of the density matrix arises from the unitary evolution of the system plus environment. Suppose that the initial state of system and environment is $|\psi\rangle_S \otimes |\psi\rangle_E$, and that we define an orthogonal basis $\{|0\rangle_E, |1\rangle_E, ..., |N-1\rangle_E\}$ of the environment. It is safe to assume $N$ finite, as we can always assume that only a finite subspace of the Hilbert space of the environment is involved in the system-environment interaction. Then the unitary evolution can be expressed as

$$\hat{U}|\psi\rangle_S \otimes |\psi\rangle_E = \sum_a^{N-1} \hat{M}_a |\psi\rangle_S \otimes |a\rangle_E \tag{10.49}$$

where $\hat{M}_a$ are the operators acting on the $\mathcal{H}$ space of the system. Notice that the vectors $\hat{M}_a|\psi\rangle_S$ are not normalized, as they contain the amplitudes associated with the expansion of $\hat{U}$. In other words

$$\hat{U}|\psi\rangle_S \otimes |\psi\rangle_E = \sum_a C_a |\phi_a\rangle_S \otimes |a\rangle_E \tag{10.50}$$

where we have defined $C_a \left|\phi_a\right\rangle_S = \hat{M}_a \left|\psi\right\rangle_S$. Since $\hat{U}$ is unitary, then

$$\begin{aligned} ||\hat{U} \left|\psi\right\rangle_S \otimes \left|0\right\rangle_E ||^2 &= 1 && (10.51)\\ &= ||\hat{M}_a \left|\psi\right\rangle_S \otimes \left|a\right\rangle_E ||^2 && (10.52)\\ &= \sum_{a,b} \left\langle\psi\right| \hat{M}_a^\dagger \hat{M}_b \left|\psi\right\rangle \left\langle a\right| \left|b\right\rangle && (10.53)\\ &= \sum_a \left\langle\psi\right| \hat{M}_a^\dagger \hat{M}_a \left|\psi\right\rangle && (10.54) \end{aligned}$$

This must be true for any $\left|\psi\right\rangle$ as the operator $\hat{M}_a$ define $\hat{U}$ and do not depend on $\left|\psi\right\rangle$. Then the relation holds:

$$\sum_a \hat{M}_a^\dagger \hat{M}_a = \mathbb{I} \tag{10.55}$$

If we now trace onto the environment, the initial state is

$$\begin{aligned} \hat{\rho}(0) &= \mathrm{Tr}_E(\left|\psi_S\right\rangle \left|0\right\rangle_E \left\langle 0\right|_E \left\langle\psi\right|_S) && (10.56)\\ &= \left|\psi\right\rangle_S \left\langle\psi\right|_S , && (10.57) \end{aligned}$$

and the state at time $t$ is

$$\begin{aligned} \hat{\rho}(t) &= \mathrm{Tr}_E\left(\left(\sum_a \hat{M}_a \left|\psi\right\rangle_S \left|a\right\rangle_E\right)\left(\sum_b \left\langle b\right|_E \left\langle\psi\right|_S \hat{M}_b^\dagger\right)\right) && (10.58)\\ &= \sum_a \hat{M}_a \left|\psi\right\rangle_S \left\langle\psi\right|_S \hat{M}_a^\dagger && (10.59)\\ &= \sum_a \hat{M}_a \hat{\rho}(0) \hat{M}_a^\dagger \equiv \mathcal{E}(\hat{\rho}(0)) && (10.60) \end{aligned}$$

We see that this is the generalization of the unitary case in which $\hat{\rho}(0) \to U\hat{\rho}(0)U^\dagger$, where $U^\dagger U = \mathbb{I}$

Notice that the map $\hat{\rho} \to \mathcal{E}(\hat{\rho})$ is linear. Therefore it also holds if the initial state is not a pure state. A linear map $\hat{\rho} \to \mathcal{E}(\hat{\rho}) = \sum_a \hat{M}_a \hat{\rho} \hat{M}_a^\dagger$ is called a *quantum channel*. Other names are *super operator* for $\mathcal{E}$, because it acts on the Liouville space of density matrices, *operator-sum representation*, or *trace-preserving completely positive map* (TPCP). The $\hat{M}_a$ operators are called Kraus operators, and the number of Kraus operators is the *Kraus number* or *Kraus rank*.

The properties of a quantum channel are:

1. Linearity: $\mathcal{E}(\alpha\hat{\rho}_1 + \beta\hat{\rho}_2) = \alpha\mathcal{E}(\hat{\rho}_1) + \beta\mathcal{E}(\hat{\rho}_2)$
2. Preserves hermiticity: $\hat{\rho} = \hat{\rho}^\dagger \implies \mathcal{E}(\hat{\rho}) = \mathcal{E}(\hat{\rho})^\dagger$
3. Preserves positivity: $\hat{\rho} \geq 0 \implies \mathcal{E}(\hat{\rho}) \geq 0$
4. Preserves trace: $Tr(\hat{\rho}) = 1 \implies Tr(\mathcal{E}(\hat{\rho})) = 1$

These properties explain the TP.P in TPCP. *Complete positivity* means that if we extend the Hilbert space $\mathcal{H}_S \to \mathcal{H}_S \otimes \mathcal{H}'$ then $\mathcal{E} \otimes \mathbb{I}$ (obtained from $\hat{M}_a \otimes \mathbb{I}$) is also positive for any possible extension $\mathcal{H}'$.

One can show that the property of complete positivity is needed for a quantum channel to represent quantum mechanical evolution and is fulfilled by the operator-sum representation. It can also be shown that any trace-preserving completely positive (TPCP) linear map is a quantum channel, i.e., it has both an operator-sum representation and a unitary representation on an appropriately extended Hilbert space. This highlights the generality of the operator-sum representation. Moreover, the operator-sum representation of a quantum channel is not unique. By changing the basis of environment states, one gets different Kraus operators for the same quantum channel: if $|a\rangle = \sum_\mu |\mu\rangle V_{\mu a}$, then the Kraus operators $\hat{N}_\mu = \sum_a V_{\mu a}\hat{M}_a$ define the same quantum channel. Another remark is that a quantum channel is generally irreversible, and is reversible if and only if it is unitary.

### 10.1.2 Noisy quantum channels

We are now at the section of interest in terms of quantum computing and introduce three quantum channels that are commonly identified as the main types of errors occurring on qubits in a quantum computer.

**Depolarizing channel**

This is the most commonly assumed error on a qubit. With probability 1-p the qubit does nothing, but with a probability of p and error occurs as follows, the error can be of three types:

1. Bit flip: $|\psi\rangle \to \hat{\sigma}_1 |\psi\rangle$ where $\hat{\sigma}_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
2. Phase flip: $|\psi\rangle \to \hat{\sigma}_3 |\psi\rangle$ where $\hat{\sigma}_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
3. Both errors: $|\psi\rangle \to \hat{\sigma}_2 |\psi\rangle$ where $\hat{\sigma}_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$

$\hat{\sigma}_j$ are Pauli matrices as we know. The unitary representation requires an environment of dimension d=4, as we know we have 4 possible outcomes.

$$\hat{U} |\psi\rangle \otimes |0\rangle = \sqrt{1-p}\, |\psi\rangle \otimes |0\rangle + \sqrt{\frac{p}{3}}(\hat{\sigma}_1 |\psi\rangle \otimes |1\rangle + \hat{\sigma}_2 |\psi\rangle \otimes |2\rangle + \hat{\sigma}_3 |\psi\rangle \otimes |3\rangle) \quad (10.61)$$

From here we can derive the operator-sum representation by tracing over the environment. We see that in general $\hat{M}_a = \langle a|_E \hat{U}$, in this case:

$$\begin{gathered} \hat{M}_0 = \sqrt{1-p}\ \mathbb{I} \\ \hat{M}_1 = \sqrt{\frac{p}{3}}\hat{\sigma}_1,\ \hat{M}_2 = \sqrt{\frac{p}{3}}\hat{\sigma}_2,\ \hat{M}_3 = \sqrt{\frac{p}{3}}\hat{\sigma}_3 \end{gathered} \quad (10.62)$$

From the properties of the Pauli matrices we easily check that $\sum_a \hat{M}_a^\dagger \hat{M}_a = \mathbb{I}$. Finally:

$$\mathcal{E}(\hat{\rho}) = (1-p)\hat{\rho} + \frac{p}{3}(\hat{\sigma}_1\hat{\rho}\hat{\sigma}_1 + \hat{\sigma}_2\hat{\rho}\hat{\sigma}_2 + \hat{\sigma}_3\hat{\rho}\hat{\sigma}_3) \quad (10.63)$$

We see that this error corresponds to assuming that on a qubit one of the three possible Pauli errors occur with probability $p/3$. The probability p transfers into an error rate for physical quantum hardware. Then, the so called digital error model is often adopted to model errors. Instead of actually evolving the density matrix of a quantum computer with $\mathcal{E}(\hat{\rho})$, one simulates the circuit by including random unitary errors $\hat{\sigma}_j$ with probability p on each qubit. The density matrix $\hat{\rho}$ of the output state of the noisy circuit is then obtained by statistically averaging over several repetitions of the circuit, each time with a random configurations of errors.

### Dephasing channel

It is also known as phase-damping channel, it is a toy model for decoherence and it can be used to explain the Schrödinger's cat paradox. Unitary representation: it needs three states in the environment, on the computational basis:

$$\begin{aligned} |0\rangle \otimes |0\rangle &\to \sqrt{1-p}\,|0\rangle \otimes |0\rangle + \sqrt{p}\,|0\rangle \otimes |1\rangle \\ |1\rangle \otimes |0\rangle &\to \sqrt{1-p}\,|1\rangle \otimes |0\rangle + \sqrt{p}\,|1\rangle \otimes |2\rangle \end{aligned} \tag{10.64}$$

Operator sum-representation:

$$\hat{M}_0 = \sqrt{1-p}\;\mathbb{I},\ \hat{M}_1 = \sqrt{p}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},\ \hat{M}_2 = \sqrt{p}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \tag{10.65}$$

We see that:

$$\hat{M}_1 = \frac{\sqrt{p}}{2}(\mathbb{I} + \hat{\sigma}_3),\quad \hat{M}_2 = \frac{\sqrt{p}}{2}(\mathbb{I} - \hat{\sigma}_3) \tag{10.66}$$

Then:

$$\mathcal{E}(\hat{\rho}) = \sum_a \hat{M}_a \hat{\rho} \hat{M}_a^\dagger = (1 - \frac{p}{2})\hat{\rho} + \frac{p}{2}\hat{\sigma}_3\hat{\rho}\hat{\sigma}_3 \tag{10.67}$$

From which we can also deduce the corresponding digital error model. Notice that if: $\rho = \begin{pmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{pmatrix}$ then $\mathcal{E}(\rho) = \begin{pmatrix} \rho_{00} & (1-p)\rho_{01} \\ (1-p)\rho_{10} & \rho_{11} \end{pmatrix}$

Suppose $p = \Gamma\Delta t << 1$ where $\Delta t$ is a time step and $\Gamma$ and error rate. Then for $t \to n\Delta t$ we have:

$$\mathcal{E}^n(\rho) = \begin{pmatrix} \rho_{00} & (1-p)^n\rho_{01} \\ (1-p)^n\rho_{10} & \rho_{11} \end{pmatrix} \tag{10.68}$$

This models the time evolution with dephasing error occurring with probability per unit time $\Gamma$. But $(1-p)^n = (1 - \frac{\Gamma t}{n})^n \xrightarrow[\Delta t \to 0]{n \to \infty} e^{-\Gamma t}$ Then:

$$\rho(t) = \begin{pmatrix} \rho_{00} & e^{-\Gamma t}\rho_{01} \\ e^{-\Gamma t}\rho_{10} & \rho_{11} \end{pmatrix} \tag{10.69}$$

We see that only the off-diagonal terms decay. Suppose we start with a schrödinger's cat state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where $|0\rangle$ represents the cat being alive and $|1\rangle$ represents the cat being dead. Then at t=0:

$$\begin{aligned} \hat{\rho}(0) &= |\psi\rangle\,\langle\psi| \\ \rho(0) &= \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix} \end{aligned} \tag{10.70}$$

Then at a later time we will have:

$$\rho(t) = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* e^{-\Gamma t} \\ \alpha^*\beta e^{-\Gamma t} & |\beta|^2 \end{pmatrix} \to \begin{pmatrix} |\alpha|^2 & 0 \\ 0 & |\beta|^2 \end{pmatrix} \tag{10.71}$$

which is just a statistical mixture of dead/alive. So decoherence transforms the crazy pure state $|\psi\rangle$ into a trivial statistical mixture, meaning by repeating the experiment, the cat was dead from the very beginning with a probability of $|\alpha|^2$ and alive with the probability of $|\beta|^2$. $\Gamma$ is the result of the interactions of all the microscopic degrees of freedom with the environment. Then $\Gamma$ is very large for large systems and decoherence occurs very fast. This is the most basic interpretation of decoherence and of why it is difficult to prepare macroscopic systems in a quantum superposition of states. However this simple picture is not complete and not conclusive.

### Amplitude damping channel

This channel describes the process in which a qubit decays from $|1\rangle$ to $|0\rangle$ by emitting some energy into the environment. Notice that this assumes some physical knowledge of the quantum computer, as we are assuming that the energies associated with $|1\rangle$ and $|0\rangle$ are different and that $|1\rangle$ and $|0\rangle$ are eigenstates of the system without external control.

The unitary representation requires two pointer states in the environment:

$$\begin{aligned} &|0\rangle_S \otimes |0\rangle_E \to |0\rangle_S \otimes |0\rangle_E \\ &|1\rangle_S \otimes |0\rangle_E \to \sqrt{1-p}\,|1\rangle_S \otimes |0\rangle_E + \sqrt{p}\,|0\rangle_S \otimes |1\rangle_E \end{aligned} \tag{10.72}$$

The corresponding Kraus operators are:

$$M_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix}, \quad M_1 = \begin{pmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{pmatrix} \tag{10.73}$$

and $\sum_a M_a^\dagger M_a = \mathbb{I}$ can be easily verified. $M_1$ describes the "jump" from $|1\rangle$ to $|0\rangle$, $M_0$ describes what happens if no jump occurs. The initial density matrix $\rho_{ij}$ varies as:

$$\rho \to \mathcal{E}(\rho) = M_0\rho M_0^\dagger + M_1\rho M_1^\dagger = \begin{pmatrix} \rho_{00} + p\rho_{11} & \sqrt{1-p}\rho_{01} \\ \sqrt{1-p}\rho_{10} & (1-p)\rho_{11} \end{pmatrix} \tag{10.74}$$

As before, set $p = \Gamma\Delta t << 1$ and $t \to n\Delta t$. Then $(1-p)^n = (1 - \frac{\Gamma t}{n})^n \xrightarrow[\Delta t \to 0]{n \to \infty} e^{-\Gamma t}$ and the same limit holds under the square root. Then, n applications of $\mathcal{E}(\rho)$ with a constant t and $n \to \infty$ gives:

$$\rho(t) = \begin{pmatrix} \rho_{00} + (1 - e^{-\Gamma t})\rho_{11} & e^{-\Gamma t/2}\rho_{01} \\ e^{-\Gamma t/2}\rho_{10} & e^{-\Gamma t}\rho_{11} \end{pmatrix} \tag{10.75}$$

The amplitude damping channel thus causes a decay of both the "coherences" (i.e. the off-diagonal terms of $\rho$) and of the "populations" (i.e. the diagonal terms) with two time constants: $T_1 = \Gamma^{-1}$ and $T_2 = 2T_1 = 2\Gamma^{-1}$. In the limit $t \to \infty$ the qubit will irreversibly decay to the state $|0\rangle$ i.e. $\rho(t) \to |0\rangle\langle 0|$.

Notice that the dephasing channel only makes the coherence decay, with a characteristic time that is usually called $T_2^*$, $T_2^*$ is the decoherence time, or pure dephasing time, as it only affects the coherences without touching the population. In a physical system usually both processes are present and one defines $T_2'$ such that $\frac{1}{T_2'} = \frac{1}{T_2} + \frac{1}{T_2^*}$

# Chapter 11

# Quantum Error Correction

## 11.1 Quantum error correction

Digital electronics and computers in particular are subject to errors. Similarly, quantum computers are subject to errors. QEC is about the theory and methods of error detection and correction in QC.

Error rates in classical modern RAM range in the order of $10^{-10}$ errors/bit/hour. (The fundamental limit to errors are cosmic rays!) These errors are mostly not corrected. They almost always affect non-sensible data (such as picture or videos) but they can indeed compromise code execution and are the major cause of crashes or data corruption in large-scale comp. facilities.

Some sensible hardware adopts Error-Correcting Code memory (ECC-RAM) which can correct (and detect) errors up to $n$ bits.

The scenario is different in QC for two reasons.

First, error rates are much higher at the time of writing. Roughly $10^{-2}$ error per 2-qubit operation, and $10^{-3}$ error per 1-qubit operation, and roughly 3% error at readout. With such an error rate, we cannot run an arbitrary algorithm successfully without an error correction strategy.

There are however algorithms specially conceived for operating on Noisy Intermediate-Scale Quantum hardware (NISQ). These algorithms can operate with errors because the output is distributed randomly around the ideal output, which can be statistically extrapolated.

An example is algorithms to compute the expectation values of Hamiltonians, used in Variational Quantum Eigensolvers (VQE).

Second, classical computers can run a parity check, i.e. read the memory and algorithmically detect which error occurred so that it can be corrected. In a quantum memory, if we read a register, it will collapse over the state that has been read out, in an irreversible way, eventually destroying the information.

QEC must thus develop ways of correcting errors in an agnostic way with respect to the actual state of the register.

## 11.2 Repetition codes

Classical ECC may use the strategy of cloning logical data onto physical memory. Consider the code:

$$0 \to 000 \quad 1 \to 111 \tag{11.1}$$

Here we will call a "code" a way in which logical data is encoded onto physical hardware. Can we build a quantum repetition code?

No. A QRC needs to clone any arbitrary quantum state $|\psi\rangle$ of a qubit:

$$|\psi\rangle \to |\psi\rangle \otimes |\psi\rangle \otimes |\psi\rangle \tag{11.2}$$

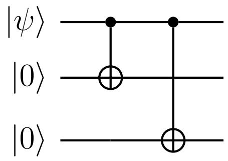
This is not possible because of the no-cloning theorem, stating that it is not possible to clone an arbitrary quantum state without any prior knowledge of what state it is. However, we can still encode the comp. basis using repetition. Take 3 qubits:

$$|0\rangle \to |0_L\rangle = |000\rangle, \quad |1\rangle \to |1_L\rangle = |111\rangle \tag{11.3}$$

This does not violate the no-cloning because:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \to \alpha|000\rangle + \beta|111\rangle \tag{11.4}$$

It can be simply realized by the following circuit:



(11.5)

This code protects against a specific error, the bit-flip error $X$. Let us define the error syndrome, i.e. observables with two important features:

1. They detect which error on which physical qubit. 2. Measuring them doesn't alter quantum info.

Introduce the four observables:

$$\begin{aligned} P_0 &= |000\rangle\langle 000| + |111\rangle\langle 111| \quad \text{no error} && (11.6)\\ P_1 &= |100\rangle\langle 100| + |011\rangle\langle 011| \quad \text{error on qubit 1} && (11.7)\\ P_2 &= |010\rangle\langle 010| + |101\rangle\langle 101| \quad \text{error on qubit 2} && (11.8)\\ P_3 &= |001\rangle\langle 001| + |110\rangle\langle 110| \quad \text{error on qubit 3} && (11.9) \end{aligned}$$

Suppose a bit flip on qubit 1 occurred. Now:

$$|\psi\rangle = \alpha|100\rangle + \beta|011\rangle \tag{11.10}$$

Notice that $P_1|\psi\rangle = |\psi\rangle$ and $\langle\psi|P_1|\psi\rangle = 1$. So if we measure $P_1$, we obtain 1 with certainty, and there is no collapse, so $\alpha$ and $\beta$ are preserved. If we measure $P_0, P_2, P_3$, then $\langle\psi|P_j|\psi\rangle = 0$.

There is no outcome and no projection, as we are measuring an observable acting on a subspace orthogonal to $|\psi\rangle$. Either way, the information is preserved.

So the QECC can measure the $P_j$'s. If $P_0 = 1$, then do nothing. If $P_j = 1$ with $j = 1, 2, 3$, then apply $X$ to the corresponding qubit. We see that:

$$X_1(\alpha|100\rangle + \beta|011\rangle) = \alpha|000\rangle + \beta|111\rangle \tag{11.11}$$

and the error is corrected without losing quantum information.

Two questions: First, can we correct $X$-errors on more than 1 qubit? Second, can we correct errors other than $X$?

QEC theory provides answers to these two questions. Let's start from the second. Notice first that the code:

$$|0\rangle \to |0_L\rangle = |+++\rangle, \quad |1\rangle \to |1_L\rangle = |---\rangle \tag{11.12}$$

where:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{11.13}$$

Protects against the $Z$ error, because $Z|+\rangle = |-\rangle$ and $Z|-\rangle = |+\rangle$, i.e. $Z$ acts on $|+\rangle$ exactly as $X$ on $|0\rangle$ and errors can be corrected in a similar way. Now, we can combine the two strategies into a 9-qubit code known as Shor's code.

$$|0\rangle \to |0_L\rangle = (|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \tag{11.14}$$

$$|1\rangle \to |1_L\rangle = (|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \tag{11.15}$$

As before, the inner repetition layer corrects signbit flip errors:

$$|010\rangle \pm |101\rangle \xrightarrow{\text{correction}} |000\rangle \pm |111\rangle \tag{11.16}$$

The outer rep. layer corrects $Z$ (i.e. phase flips) errors. This is done by measuring the phase and taking the majority of three copies:

$$(|.\rangle + |.\rangle)(|.\rangle - |.\rangle)(|.\rangle + |.\rangle) \xrightarrow{\text{correction}} (|.\rangle + |.\rangle)(|.\rangle + |.\rangle)(|.\rangle + |.\rangle) \tag{11.17}$$

Error syndromes can be defined and corresponding error correction operations found.

Notice that the effect of a phase flip error on any of the 1st 3 qubits (or of any of the 3 qubits in the same group), is the same:

$$|000\rangle \pm |111\rangle \to |000\rangle \mp |111\rangle \tag{11.18}$$

This is OK because also the syndrome and error correction can be the same. For example, a recovery op. would be $Z_1, Z_2, Z_3$.

Now, this QECC allows to detect and correct also an $X$ and a $Z$ error on the same qubit, as can be easily verified. Therefore, an error of the type $X_j Z_j$ can be corrected.

The following *theorem* also holds: If a QECC can correct errors $A$ and $B$, then it can also correct any linear combination of $A$ and $B$.

Indeed, if the code can correct $A$ and $B$, then there must be two error syndromes $S_A$ and $S_B$ such that they are measured with certainty on $A|\Psi\rangle$ and $B|\Psi\rangle$, respectively. Then, if the error $aA + bB$ occurs:

$$|\Psi\rangle \to aA|\Psi\rangle + bB|\Psi\rangle \tag{11.19}$$

Measuring either $S_A$ or $S_B$ will collapse the state on $A|\Psi\rangle$ or $B|\Psi\rangle$, resp., because $S_A$ and $S_B$ are orthogonal: $S_A$ is zero on $B|\Psi\rangle$ and $S_B$ is zero on $A|\Psi\rangle$.

After the collapse, we are left with one of the two errors, and we know which one it is, so we can restore it using the corresponding recovery operator. An example is indeed $A = X$ and $B = Z$, because op. $X$ and $Z$ on the same qubit bring the code to orthogonal states: on the 1st qubit,

$$(|000\rangle \pm |111\rangle)()() \xrightarrow{X} (|100\rangle \pm |011\rangle)()() \tag{11.20}$$

$$(|000\rangle \pm |111\rangle)()() \xrightarrow{Z} (|000\rangle \mp |111\rangle)()() \tag{11.21}$$

The power of QECC resides in this theorem which ultimately stems from the linearity of quantum mechanics. Let us summarize:

If we can correct $X$ and $Z$ on a qubit, we can also correct $XZ$ and $ZX$. We can also correct any linear comb. of $X, Z, XZ, ZX$, and $I$. But we see that $iXZ = Y$ and that $\{I, X, Y, Z\}$ are a basis of all self-adjoint ops on a single qubit. In particular,

$$R(\theta, \hat{n}) = e^{-i\theta\vec{n}\cdot\vec{\sigma}} \tag{11.22}$$

where $\vec{\sigma} = \{\hat{X}, \hat{Y}, \hat{Z}\}$, is the most general unitary op. on 1 qubit. Taylor expansion and the fact that $X^2 = Y^2 = Z^2 = I$ show that $R(\theta, \hat{n})$ is always a linear combination of $I, X, Y, Z$.

We conclude that Shor's code, and in general a code that can correct $X$ and $Z$, can correct any 1-qubit error.

The argument extends to linear comb. of errors on single-but different qubits: if one can correct $I \otimes A \otimes I \otimes ... \otimes I$ and $I \otimes B \otimes I \otimes ... \otimes I$, then also their linear comb. can be corrected.

This makes it possible to correct to some extent small errors on multiple qubits. Suppose you have an error:

$$A = \otimes(I + \epsilon E_j) \tag{11.23}$$

with a small $\epsilon$:

$$A = I + \epsilon(E_1 \otimes I \otimes ... + I \otimes E_2 \otimes I + ...) + O(\epsilon^2) \tag{11.24}$$

So to order $\epsilon$, we can correct also small multiple errors. However, single-qubit QECC can't correct, in general, multiple qubit errors such as, e.g., $X_j Z_k$ with $j \neq k$.

A QECC that can correct such products, can by linearity correct all 2-qubit errors, including entangling (i.e. non-separable) errors. Notice that all tensor products of $I \times X, Y, Z$, with additional factors $\pm I$ and $\pm i$, form the Pauli group $P_n$, which is important for QECC.

This may all seem a bit confusing. It can however be generalized into a very neat theorem:

## 11.3 Knill-Laflamme Theorem

Given a code $C$ (i.e. a subspace of the Hilbert space of dim $2^n$ to encode $n$-qubits) and a group of errors $S$, $C$ detects/corrects $S$ if and only if, for any pair of orthogonal codewords $|\Phi\rangle, |\Psi\rangle$ with $\langle\Phi|\Psi\rangle = 0$,

The two codewords must be orthogonal (i.e. distinguishable) also after errors. This means that for any $E_r, E_s \in S$:

$$\langle E_r\Phi|E_s\Psi\rangle = \langle\Phi|E_r^\dagger E_s|\Psi\rangle = 0 \tag{11.25}$$

If one considers an orthogonal basis $\{|\Psi_1\rangle, ..., |\Psi_k\rangle\}$ of $C$, then the Knill-Laflamme condition translates into:

$$\langle\Psi_i|E_r^\dagger E_s|\Psi_j\rangle = \delta_{ij} C_{rs} \quad \forall E_r, E_s \in S \tag{11.26}$$

where $C_{rs}$ does not depend on $i, j$. If $E_r, E_s$ are chosen from an orthogonal basis of $S$, then $C_{rs}$ is a hermitian matrix. We will discuss now the proof and its meaning.

### 11.3.1 Proof and Discussion

Consider a code $C$, i.e., a space of dim $2^n$. Consider two orthogonal states $|\Phi\rangle$ and $|\Psi\rangle$ within the code. We may understand $|\Phi\rangle$ and $|\Psi\rangle$ as two states of the comp. basis, or just two arbitrary states.

The basic requirement of QECC is that if an error occurs, we must detect it. We may also say that we *must* be able to distinguish it, but there are exceptions to this, as we will see, for the states of the comp. basis.

However, for an arbitrary state $|X\rangle = \alpha|\Phi\rangle + \beta|\Psi\rangle$ with $|\alpha|^2 + |\beta|^2 = 1$, and given two errors (that are corrected by the code) $E$ and $F$, we must be able to distinguish the two errors. This means we must be able to distinguish the two states $E|X\rangle$ and $F|X\rangle$. For arbitrary states, it is only possible to distinguish them with certainty if they are orthogonal. Thus, we require:

$$\langle X|E^\dagger F|X\rangle = 0 \quad \forall \alpha, \beta \in C \tag{11.27}$$

This means:

$$|\alpha|^2\langle\Phi|E^\dagger F|\Phi\rangle + |\beta|^2\langle\Psi|E^\dagger F|\Psi\rangle + \alpha\beta^*\langle\Phi|E^\dagger F|\Psi\rangle + \alpha^*\beta\langle\Psi|E^\dagger F|\Phi\rangle = 0 \tag{11.28}$$

For this to be true $\forall\alpha,\beta$, we need the two products in the second line to be zero:

1. $\langle\Phi|E^\dagger F|\Psi\rangle = 0 \quad \forall|\Phi\rangle, |\Psi\rangle$ such that $\langle\Phi|\Psi\rangle = 0$,
2. $\langle\Phi|EF|\Phi\rangle = 0 \quad \forall|\Phi\rangle$.

The first line would suggest that also:

$$\langle\Phi|EF|\Phi\rangle = 0 \quad \forall|\Phi\rangle \tag{11.29}$$

and this seems to repeat the initial requirement $\langle X|E^\dagger F|X\rangle = 0$.

This second condition is sufficient but not necessary. In fact, in the Shor code we have seen that a $Z$-error on each of the three qubits in the same group will give just a "-" sign, thus leading to the same state:

$$|000\rangle + |111\rangle \rightarrow |000\rangle - |111\rangle \tag{11.30}$$

The code is then said to be *degenerate*. Thus, we can accept that the correction can be made on *any one* of the three qubits.

In conclusion, if $E$ and $F$ must be distinguished, then (1) and (2) are necessary. If $E$ and $F$ can be corrected indistinctively, as for the $Z$-error in Shor's code, then we may allow:

$$\langle X|E^\dagger F|X\rangle \neq 0 \tag{11.31}$$

However, condition (1) must still hold. If (1) does not hold, then it may build an error $G = \alpha E + \beta F$. We know that a code must be linear, so correct all lin. comb. of errors. Therefore, $|\Phi\rangle$ and $|\Psi\rangle$ must be orthogonal also after the same error $G$:

$$\langle\Phi|G^\dagger G|\Psi\rangle = 0 \quad \forall\alpha,\beta \tag{11.32}$$

But since $G = \alpha E + \beta F$, this is only possible if condition (1) is fulfilled.

Finally, we see that another condition for QECC is:

$$\langle\Psi|E^\dagger E|\Psi\rangle = \langle\Phi|E^\dagger E|\Phi\rangle \tag{11.33}$$

This condition is required because we want to be able to revert the error. We require:

$$E^\dagger E|X\rangle = e^{i\theta}|X\rangle \tag{11.34}$$

Recall that $E$ is not necessarily unitary.

In general, for a noisy quantum channel:

$$\mathcal{E}(\rho) = \sum E_j\rho E_j^\dagger \tag{11.35}$$

we only have:

$$\sum E_j^\dagger E_j = I \tag{11.36}$$

So the above condition is not obvious. Again, if:

$$E^\dagger E|X\rangle = e^{i\theta}|X\rangle, \tag{11.37}$$

then:

$$\langle X|E^\dagger E|X\rangle = e^{-i\theta}. \tag{11.38}$$

$$= |\alpha|^2\langle\Phi|E^\dagger E|\Phi\rangle + |\beta|^2\langle\Psi|E^\dagger E|\Psi\rangle \tag{11.39}$$

(The cross terms are zero because of the other condition.) But since $\forall\alpha,\beta$ we have $|\alpha|^2 + |\beta|^2 = 1$, the only way to fulfill this requirement is that:

$$\langle\Phi|E^\dagger E|\Phi\rangle = \langle\Psi|E^\dagger E|\Psi\rangle. \quad \square \tag{11.40}$$

Similar considerations prove the orthogonal version of the theorem, starting from the general one:

$$\langle\Psi_i|E_r^\dagger E_s|\Psi_j\rangle = \delta_{ij}C_{rs}, \tag{11.41}$$

where $C_{rs}$ is a hermitian matrix if $E_s$ are taken from an orthogonal basis of generators. If the code is non-degenerate, then $C_{rs}$ has minimum rank.

The Knill-Laflamme theorem is of fundamental importance for the design of QECCs.

## 11.4 Bounds on the Parameters of a QECC

We can set a bound on the parameters of a QECC. Each error $E$ acting on each basis codeword $|\Psi_i\rangle$ must produce a linearly independent state (i.e., orthogonal). All these states must fit into the Hilbert space of the full system of $n$ physical qubits, of dimension $2^n$.

If the code encodes $k$ qubits and corrects up to $t$-qubit errors, then the following *quantum Hamming bound* holds (for non-degenerate codes):

$$\sum_{j=0}^{t} 3^j \binom{n}{j} 2^k \leq 2^n \tag{11.42}$$

where the number between parentheses is the number of errors of weight $t$ or less. The ”weight” of an error is the number of single qubit errors in the tensor product:

$$I \otimes \cdots \otimes I \otimes E_2 \otimes I \otimes \cdots \otimes I \tag{11.43}$$

has weight 2.

For $t = 1, k = 1$, we see that $n \geq 5$. Indeed, there is an $n = 5$ code that we will introduce later.

If a code corrects $t$ errors, then it is said to have distance:

$$d = 2t + 1. \tag{11.44}$$

Distance means that it takes $d$ single qubit changes to go from one codeword to another. A code using $n$ qubits to encode $k$ qubits with distance $d$, is denoted as $[[n, k, d]]$. The double brackets are to distinguish the notation from that for classical ECC. Shor's code is a $[[9, 1, 3]]$ code.

Another bound is on the existence of the code. A code exists if:

$$\sum_{j=0}^{d-1} 3^j \binom{n}{j} 2^k \leq 2^n. \tag{11.45}$$

Notice that $d - 1 = 2t$, so this bound is more restrictive than the other. Finally, the Knill-Laflamme bound requires:

$$n - k \geq 2d - 2. \tag{11.46}$$

These last two bounds are given without proof.

We introduce now the notion of *stabilizer*, which makes the construction of QECC formal and systematic.

## 11.5 The Stabilizer Formalism

To understand the stabilizer formalism, let us first review the Shor code. How do we actually measure the 4 projectors?

It can be done by parity measurement. More specifically, take the first 3 qubits. By measuring the observable:

$$Z \otimes Z \otimes I \tag{11.47}$$

we can detect if a bit flip has occurred on the first two qubits. In fact, if the two qubits are 00 or 11, the measure gives +1, and if they are different, the measure gives −1. Same for qubit 2 and 3, by measuring:

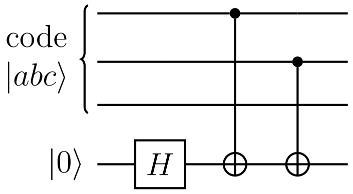
$$I \otimes Z \otimes Z. \tag{11.48}$$

Notice that the syndrome operators anticommute with the error operators. For example:

$$\{Z \otimes Z \otimes I, X \otimes I \otimes I\} = \{Z \otimes Z \otimes I, I \otimes X \otimes I\} = 0. \tag{11.49}$$

The two measurements tell us on which of the three qubits the error occurred. The measurement can be carried out with C-Z's and one ancilla qubit.

For example, for $Z \otimes Z \otimes I$, we will use the circuit:



(11.50)

$$\sum_{abc} C_{abc}(|0\rangle\langle abc| + (-1)^{c00b}|1\rangle\langle abc|) = \sum_{abc} C_{abc}(|0\rangle + (-1)^{a\oplus b}|1\rangle)\langle abc|. \tag{11.51}$$

Now it is enough to measure on the $|+\rangle$ basis the ancilla qubit (first go to the comp. basis with another Hadamard, then measure in the comp. basis) to know $a \oplus b$, which is the parity of $ab$.

Notice that the anticommutation rule between the syndrome operator like $Z \otimes Z \otimes I$ and the error operator $X \otimes I \otimes I$ or $I \otimes X \otimes Z$ is the essential feature to detect the error.

If $M$ is the syndrome and $E$ the error, then on a codeword $|\Psi\rangle$ we have:

$$ME|\Psi\rangle = -EM|\Psi\rangle = -E|\Psi\rangle \tag{11.52}$$

because the codeword is per definition an eigenvector of $M$ with eigenvalue $+1$ (no error detected). So $M$ is built in order to be the identity within the code space and *not* the identity outside.

Similarly, for Shor's code, to detect a $Z$-error, we must check if the signs of the three qubits are all the same. For this we can measure:

$$X\otimes X\otimes I\otimes X\otimes X\otimes I\otimes I\otimes I\otimes I \quad \text{and} \quad I\otimes I\otimes I\otimes X\otimes X\otimes X\otimes X\otimes I\otimes I. \tag{11.53}$$

In fact, if one $Z$-error occurs in the 1st or 2nd group of three qubits, then:

$$|000\rangle + |111\rangle \to |000\rangle - |111\rangle. \tag{11.54}$$

In the code, both groups must have the same sign, so if they are instead opposite, then one of the two $X^{\otimes 3}$ will give a $+1$ factor and the other a $-1$, for a total of a $-1$ factor, thus signaling the error.

In total, we have 8 error syndromes:

$$\begin{array}{ccccccccc}
Z & Z & I & I & I & I & I & I & I \\
I & Z & Z & I & I & I & I & I & I \\
I & I & I & Z & Z & I & I & I & I \\
I & I & I & I & Z & Z & I & I & I \\
I & I & I & I & I & I & Z & Z & I \\
X & X & X & I & I & I & I & Z & Z \\
X & X & X & X & X & X & I & I & I \\
I & I & I & X & X & X & X & X & X
\end{array} \tag{11.55}$$

Again, each of these generators anticommutes with the corresponding error detected.

One error $E$ commutes with a particular set of syndromes $\{M\}$. Then measuring $-1$ on exactly all of the $M$'s in $\{M\}$ tells us that exactly that error occurred. Sometimes, it is *not* possible to tell which error occurred, like for errors $Z_1$ and $Z_2$ in Shor's code: $Z_1|14\rangle = Z_2|14\rangle$ for all codewords. This is not important because also the correction can be carried out indistinctly on bit 1 or 2. This is an example of a degenerate code.

The $S$ operators above generate an Abelian group called the "Stabilizer" of the code. The stabilizer contains all the operators $M$ in the Pauli group, for which $M|14\rangle = |14\rangle$ for all $|14\rangle$ in the code.

Recall what a Pauli group is: The $n$-qubit Pauli group $P_n$ is the group formed by all $4^n$ operators on $n$-qubits that are formed as tensor products of $I, X, Y$, or $Z$, for example $I \otimes X \otimes Z \otimes I \otimes X$ for 5 qubits, with an overall phase factor of $\pm 1$ or $\pm i$.

Demonstrate that $P_n$ is a group (the $\pm 1$ and $\pm i$ phases are required for $P_n$ to be a group). $P_n$ is not Abelian. More precisely, any pair of elements $M, N \in P_n$ either commute or anti-commute, i.e.,

$$[M, N] = MN - NM = 0 \quad \text{or} \quad \{M, N\} = MN + NM = 0. \tag{11.56}$$

Also, for each $M \in P_n$, $M^2 = \pm I$. We only need to work with $M \in P_n$ such that $M^2 = I$, but the other elements (those with the phase $\pm i$) are needed to provide the group property.

We further define the "weight" of an operator $M \in P_n$ as the number of tensor factors different from $I$. For example, $X \otimes Y \otimes I$ has weight 2.

Conversely, given an Abelian subgroup $S \subset P_n$, we can define a QECC $T(S)$ as the set of all states $|14\rangle$ such that $M|14\rangle = |14\rangle \quad \forall M \in S$.

The stabilizer $S$ must be Abelian and cannot contain $-I$. Proof: If $M, N \in S$, then per definition,

$$MN|14\rangle = M|14\rangle = |14\rangle \quad \text{and} \quad NM|14\rangle = N|14\rangle = |14\rangle. \tag{11.57}$$

So within the subspace $T(S)$, we have:

$$[M, N]|14\rangle = 0 \tag{11.58}$$

but elements of the Pauli group $P_n$ either commute or anti-commute. Then, if $M, N$ commute on a subspace, they must necessarily commute everywhere in the full Hilbert space.

Clearly, if $M = -I \in S$, there is no $|14\rangle$ such that $M|14\rangle = |14\rangle$.

Given a stabilizer with these properties, there is a non-trivial subspace whose states $|14\rangle$ are "stabilized", i.e., $M|14\rangle = |14\rangle \quad \forall M \in S$.

Let us investigate the properties of stab codes. Recall the Knill-Laflamme condition for QECC. Given a codespace $T(S)$ with basis $\{|\psi\rangle\}$ and correctable errors $\{E_r\}$, then:

$$\langle\psi|E_r^\dagger E_s|\psi\rangle = \delta_{ij} C_{rs}, \quad C_{rs} \text{ indep. of } ij. \tag{11.59}$$

As we have seen, two errors $E$ and $F$ are correctable if (1) they act the same on codewords or (2) they send $T(S)$ on two disjoint orthogonal subspaces.

(1) $E|14\rangle = F|14\rangle \quad \forall |14\rangle \in T(S)$

$$\implies EF|14\rangle = |14\rangle \implies EF \in S. \tag{11.60}$$

This is the case of $Z_1$ and $Z_2$ errors on Shor's code. They can be corrected by the same operation even if we don't know which one occurred. The code is degenerate.

(2) $E$ and $F$ send $T(S)$ into two orth. subspaces. $E|14\rangle$ and $F|14\rangle$ are orth. if $|14\rangle \in T(S)$,

$$\langle\psi|E^\dagger F|\psi\rangle = 0. \tag{11.61}$$

But if $E$ and $F$ are correctable, then there is at least one generator $M \in S$ such that

$$ME|14\rangle = \pm E|14\rangle \quad \text{and} \quad MF|14\rangle = \pm F|14\rangle. \tag{11.62}$$

i.e.,

$$\{M, E\} = [M, F] = 0 \quad \text{or} \quad \{M, F\} = [M, E] = 0. \tag{11.63}$$

Then: (suppose $\{M, E\} = 0$)

$$0 = \langle\psi|E^\dagger F|\psi\rangle = \langle\psi|ME^\dagger F|\psi\rangle = -\langle\psi|E^\dagger FM|\psi\rangle \tag{11.64}$$

$$= -\langle\psi|E^\dagger FM|\psi\rangle = -\langle\psi|E^\dagger FM|\psi\rangle \implies \{M, E^\dagger F\} = 0. \tag{11.65}$$

Conversely, $E, F$ are correctable if either $EF \in S$ or $\exists M \in S$ s.t. $\{M, EF\} = 0$.

Let us now count the states. We have $2^n$ states in the full Hilbert space of $n$-qubits. Suppose there are $a$ generators in $S$. Generators have eigenvalues $\pm 1$ in equal number, because $M \in S \implies Tr(M) = 0$ (they are Pauli operators).

$T(S)$ is eigenspace of $S$ with eigenvalue $+1$. All other orthogonal subspaces have eigenvalue $+1$ for some $M$'s and $-1$ for some others.

(Because $\{M, E\} = 0$ or $[M, E] = 0$, the subspace $ET(S)$ has constant eigenvalue for each given $M$).

Then we can have $2^a$ distinct "error syndromes", i.e., sets of $\pm 1$ eigenvalues for all $M \in S$. So we have $2^a$ distinct subspaces which must fit into the total space of dim $2^n$. We know that all these subspaces exist because for each error syndrome $(+1, -1, \ldots, +1, -1)$.

There is at least one element of the Pauli group $P_n$ that has that syndrome. Then

$$\dim(T(S)) = 2^n/2^a = 2^{n-a} = 2^k \tag{11.66}$$

Hence a stabilizer $S$ on $n$ physical qubits, with $a$ generators, corrects errors on $k = n - a$ logical qubits.

For example, Shor's code has $a = 8$ generators and $n = 9$ qubits, so it fully corrects errors on $k = 1$ logical qubit as we have seen.

What about the distance $d$ of a stabilizer code? We know that $t = d - 1$ is the maximum weight of Pauli operators for which all operators with that weight are corrected.

There is not a simple formula for $d$ apart from the bounds that we have already seen. In particular:

$$n - k \geq 2d - 2 \quad \text{(also for deg. codes)} \tag{11.67}$$

For Shor,

$$2d \leq n - k + 2 \implies d \leq \frac{n - k + 1}{2} = 5 \tag{11.68}$$

We have $d = 3$.

Until now, our analysis has identified two types of elements of the Pauli group $P_n$:

1. $M \in P_n$ such that $M \in S$ (generators of $S$).

2. $E \in P_n$ such that $E \notin S$ and there is at least one $M \in S$ such that $\{M, E\} = 0$. These are the correctable errors.

There is obviously a third subset of $P_n$:

3. $G \in P_n$ such that $G \notin S$ but $\forall M \in S, [G, M] = 0$. There are not generators and not correctable errors.

Notice that $\forall \left|\Psi\right\rangle \in T(S)$ and $\forall M \in S$,

$$MG \left|\Psi\right\rangle = GM \left|\Psi\right\rangle = G \left|\Psi\right\rangle . \tag{11.69}$$

Therefore, $G \left|\Psi\right\rangle \in T(S)$.

But it can't be that $G \left|\Psi\right\rangle = \left|\Psi\right\rangle \ \forall \left|\Psi\right\rangle$, otherwise $G$ would be in $S$.

We conclude that $G$ is a nontrivial operation acting on the codespace.

Given a subset $S$ of a group $P$, the centralizer $C(S)$ is the set of elements $g$ of $P$ such that

$$[g, M] = 0 \quad \forall M \in S. \tag{11.70}$$

The normalizer $N(S)$ obeys a weaker condition that

$$gS = Sg, \text{ i.e., } \forall M \in S, gM = Ng \text{ with } N \in S. \tag{11.71}$$

For the Pauli group, for which elements either commute or anticommute, $N(S)$ and $C(S)$ coincide.

Proof: if $gM = Ng$, then $g^{-1}Ng = M$ but $g^{-1}Ng = \pm g^{-1}gN = \pm N$. Now, if $N \in S$ then $-N = (-I)N \in S$ because $-I \in S$. Thus, $g^{-1}Ng = \pm N$.

So $N(S)$ for a stabilizer code $S$ contains all $g$ that commute with all elements of $S$, including the elements of $S$ itself.

Then $N(S) - S$ is the set of all nontrivial operators on codewords.

Notice that if $g \in N(S)$ and $M \in S$, then $h = gM \in N(S)$ as well. And:

$$h \left|\Psi\right\rangle = gM \left|\Psi\right\rangle = g \left|\Psi\right\rangle , \quad \forall \left|\Psi\right\rangle \in T(S). \tag{11.72}$$

All operators $h = gM$ form a coset of $S$ in $N(S)$ (left or right coset, it doesn't matter because $gM = Mg$).

So all operators $h$ in the same coset of $S$ in $N(S)$ act as the very same Pauli operation on codewords.

Choosing one such element $h$ from each distinct coset (they form equivalence classes), gives a set of Pauli operators on the code, i.e., defines the Pauli group $P_k$.

In group theory, this corresponds to defining the factor group:

$$N(S)/S \quad \text{which is the set of cosets of } S \text{ in } N(S). \tag{11.73}$$

Its order (No. of elements) is $4^k$.

Let us now just introduce two well-known stabilizer codes for $k = 1$, i.e., 1 qubit.

### 11.5.1 Steane code

$$\begin{matrix} Z & Z & Z & Z & I & I & I \\ Z & Z & I & Z & Z & I & I \\ Z & I & Z & I & Z & I & Z \\ X & X & X & X & I & I & I \\ X & X & I & I & X & X & I \\ X & I & X & I & X & I & X \end{matrix} \tag{11.74}$$

It has 6 generators on 7 qubits, so it encodes 1 qubit. It is a $[\![7,1,3]\!]$ code.

### 11.5.2 Calderbank-Shor-Steane code

Between classical and quantum ECC known as Calderbank-Shor-Steane construction or CSS constr. For each classical ECC, it is then possible to build a QECC using the CSS constr. A non-CSS code which is even smaller is the $[\![5,1,3]\!]$ code, which is the minimal code fulfilling the bounds given previously. The stabilizer is:

$$\begin{matrix} X & Z & Z & X & I \\ I & X & Z & Z & X \\ X & I & X & Z & Z \\ Z & X & I & X & Z \end{matrix} \tag{11.75}$$

We may now ask the question: why is it so hard to build an error-proof quantum computer? Isn't it sufficient to use one of these codes to encode logical qubits?

Let us put aside the fact that we may want to correct errors with $k > 1$ and $d > 3$. The problems are two:

1. To actually implement a code, we need to execute some quantum circuits. Are we adding more errors?
2. Once we have the QECC, how do we encode logical non-Pauli quantum gates on it?

These questions introduce the topic of fault-tolerant quantum computing.

# Chapter 12

# Fault Tolerant Quantum Computing

## 12.1 Fault Tolerance

We conclude our overview of QECC with the notion of fault tolerance and the threshold theorems. For this, we need to introduce a very important class of quantum gates: the so-called Clifford group.

The Clifford group plays a fundamental role in quantum computing for several reasons:

- Clifford gates are sufficient to encode and decode stabilizer codes.
- Clifford gates can be "easily" quantum-error-corrected via the idea of "transversal" gate encoding.
- The Gottesman-Knill theorem poses that circuits made of Clifford gates only can be simulated classically with polynomial complexity.
- More recent works have shown that the exponential complexity of quantum circuits only depends on the amount of non-Clifford gates present in the circuit. More precisely, if a circuit contains $t$ non-Clifford gates, then its classical emulation has complexity $O(2^{\alpha t} t^3)$ (see S. Bravyi, arXiv:1601.07601).
- Circuits made of Clifford gates only can produce highly entangled states, yet can be simulated efficiently using classical algorithms. This casts light on the very subtle role of entanglement in determining the exponential speedup of some quantum algorithms.

## 12.2 Clifford group

The Clifford group for $n$ physical qubits is defined as:

$$C_n = \left\{ U \in U(2^n) \,\text{s.t.}\, UPU^\dagger \in \mathcal{P}_n \,\forall P \in \mathcal{P}_n \right\} \tag{12.1}$$

where $\mathcal{P}_n$ is the Pauli group for $n$ qubits. That is, the Clifford group is the normalizer of the Pauli group $\mathcal{P}_n$ in the group of unitary operators $U(2^n)$.

Examples of single-qubit operators in $C_n$ are:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{12.2}$$

For example, take $X$ and $Z$ in $\mathcal{P}_n$:

$$HXH^\dagger = Z, \quad HZH^\dagger = X. \tag{12.3}$$

Notice that this is true also for $X \otimes X$ in $\mathcal{P}_n$:

$$H \otimes H \, X \otimes X \, H \otimes H = Z \otimes X. \tag{12.4}$$

The operation $UPU^\dagger$ is a conjugation operation and it induces a homomorphism of the group:

$$UABU^\dagger = (UAU^\dagger)(UBU^\dagger). \tag{12.5}$$

Then, for example, since $Y = iXZ$, we deduce that:

$$UYU^\dagger = iUXU^\dagger UZU^\dagger = -Y. \tag{12.6}$$

So in order to specify a Clifford operator $U$, it is sufficient to indicate its action on the generators of $\mathcal{P}_n$, i.e., on $X$ and $Z$ on each of the $n$ qubits.

In fact, the Clifford group is generated by the three operators above, $H$, $S$, and $CNOT$ (given without proof).

As already mentioned, for Clifford gates the following theorem holds:

## 12.3 Gottesman-Knill Theorem

Any circuit on $n$ qubits, made of Clifford gates, with a simple initial state (i.e., computational basis) and with final Pauli measurement, can be simulated classically in poly($n$) time.

## 12.4 Fault Tolerance in Gates, State Prep., and Measurement

We have seen that it is possible to detect and correct errors on sets of logical qubits, thanks to QECC (and stabilizer codes in particular). However, to detect and correct needs additional qubits and gates which will eventually introduce new errors. Also, we still don't know how exactly to carry out gates on logical qubits in a QECC in a safe way, without making errors worse (propagating errors).

The goal of fault tolerance is to develop a full fault-tolerant protocol that makes the processes of (1) error detection and correction, (2) applying gates on logical qubits, (3) preparing initial states, and (4) measuring final states, resilient to some amount of errors occurring on physical qubits. In this way, we can take a quantum circuit designed to work in the absence of errors and write a new circuit based on QECC, which produces the exact same output, provided the error rate on the physical qubits is not too large.

One big obstacle is error propagation. Even if gates on codewords are perfect, an uncorrected error changes once a gate $U$ is applied:

$$UE|\psi\rangle = UEU^{\dagger}U|\psi\rangle. \tag{12.7}$$

So the error is changed from $E$ to $UEU^{\dagger}$ after the gate is applied. This illustrates why the Clifford group is important. If $U \in C_n$, then for $E \in \mathcal{P}_n$, $UEU^{\dagger}$ is also an element of $\mathcal{P}_n$, so it is another error.

If $U$ is a tensor product of 1-qubit gates, then this doesn't increase the weight of the error:

$$(U_1 \otimes U_2)(E_1 \otimes I)(U_1^{\dagger} \otimes U_2^{\dagger}) = (U_1 E_1 U_1^{\dagger}) \otimes I. \tag{12.8}$$

However, if $U$ is a non-separable 2-qubit operator, then the error will propagate. This happens typically with the $CNOT$:

$$CNOT : X \otimes I \to X \otimes X, \quad CNOT : I \otimes Z \to Z \otimes Z. \tag{12.9}$$

So if a QECC implies that CNOTs can be applied on physical qubits (and this will always be the case, we can't have a quantum computing paradigm made only of separable operations), this may increase the weight of an error beyond the weight $t$ that can be corrected by the QECC.

Here we will focus on *fault-tolerant protocol* for applying gates to logical qubits. Similar considerations can be used for the other steps, i.e., state preparation, error detection and correction, and measurement.

Our scope is just to give an example of the challenges and solutions, while for a full account of QECC we refer to more specific works and courses.

## 12.5 Transversal gates

A fundamental notion in a QECC protocol is that of *transversal gates.*

Recall that for a stabilizer code $[[n, k, d]]$ with stabilizer $S$, the quotient group $N(S)/S$ contains all logical Pauli gates $P_L$ on logical qubits. Recall also that $N(S)/S$ is the set defined as:

$$N(S)/S = \{aS, bS, \dots | a, b, \in N(S)\} \tag{12.10}$$

where $\{aS\} = \{aM, aM_2, \dots | M, M_2, \dots \in S\}$.

So, a logical Pauli gate on the $k$ logical qubits can be chosen among the elements of $N(S)/S$ where not only one but all the operations in one coset act as the same Pauli gate on $T(k)$.

Suppose, for example, we want to perform a logical $\bar{Z}$ on one of the $k$-qubits (from now on we will denote logical operations on logical codewords by a bar). We choose one of the operations in the corresponding coset $Q$. Now $Q \in P_n$, so to execute it, we need to execute some Pauli gates (or $I$) on each of the $n$ physical qubits in the code.

Since we apply single-qubit Pauli gates, there is no possibility for an error to propagate. If there are $m$-Pauli errors in the $k$-qubit code before the application of $Q$, there will be $m$ Pauli errors on the same physical qubits after applying $Q$. This is only true because $Z$ is a Clifford operation, and the corresponding $Q$ is also a Clifford operation, and is a tensor product of single-qubit Clifford gates.

We have already seen in the last problem set that $Z$ on the five-qubit code can be applied by applying $Z$ on all five qubits. You can show as an exercise that, for example, applying $H$ on Shor's 7-qubit code is achieved by applying $H^{\otimes 7}$. The logical operation is not necessarily the product of the same physical operation on all qubits. For example, always for Shor's code, the logical $S$ is achieved by applying $S^\dagger$ on all physical qubits.

The notion of transversal gates arises once we try to extend this protocol to multiple-qubit logical gates. Suppose $\bar{U}$ is a $m$-qubit gate. The corresponding $\bar{U}$ may propagate errors within one block of $k$-qubits of a code.

The gate $\bar{U}$ is said to be *transversal* if it can be written as

$$\bar{U} = \bigotimes_i U_i \tag{12.11}$$

where each $U_i$ is a $m$-qubit gate on physical qubits, acting on the $i$-th qubit of each block of the code. The $U_i$ do not need to be the same.

Let us think of a simple example. Consider a $k = 1$ code $T_1(S)$, and consider two logical qubits, encoded in $T_1(S)$ and $T_2(S)$:

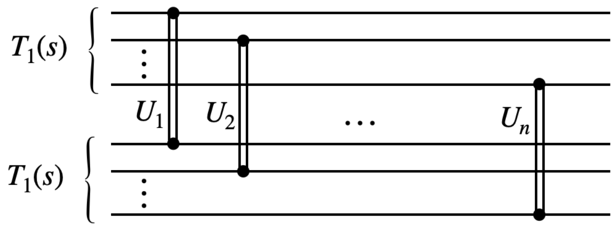


Figure 12.1: An illustration of a transversal gate.

As already mentioned, the operators $U_i$ do not need to be all the same, even though most transversal gate constructions are indeed repetitions of identical gates.

In this transversal construction, if one error is present in, e.g., bit 1 of $T_1(S)$, then $U_i$ will propagate the error to bit 1 of $T_2(S)$. But this will not increase the weight of the error within $T_1(S)$ or $T_2(S)$.

Even if errors are already present on bit 1 of both $T_1$ and $T_2$, this construction will leave the number of errors in each logical qubit code $T_i(S)$ unchanged. In this way, errors can be corrected also after the application of $\bar{U}$.

It is not possible to generalize the notion of transversal gates to gates acting on 1 qubit only in each block, without the qubits being the same (first with first, etc.). This is because, as one can easily verify, only by pairing $i$-th gate with $i$-th gate, we have that the product of two transversal gates is still transversal.

As an example, applying $\mathrm{CN\bar{O}T}$ to two logical qubits encoded in two 7-qubit blocks, can be achieved simply as

$$\mathrm{CN\bar{O}T} = \mathrm{CNOT}^{\otimes 7} \tag{12.12}$$

The importance of Clifford gates is that in most stabilizer codes they admit logical transversal encoding. For example, for the 7-qubit code, $H$, $S$, and CNOT are transversal. Since products of transversal gates are transversal, this means that the whole Clifford group can be implemented transversally.

The problem often arises with the $T$-gate, which cannot be simply implemented transversally in most cases. In fact, one can prove that a QECC with transversally encoded universal set of gates is impossible (see B. Eastin, E. Knill, arXiv:0811.4262).

A possible way to overcome the Eastin-Knill theorem is to add external resources to the QECC, such as ancilla qubits bearing customarily prepared quantum states (magic state distillation) or to avoid 2-qubit gates with the paradigm of measurement-based (one-way) quantum computing.

Finding a simple way to realize a universal set of fault-tolerant quantum gates remains, however, an open problem and a challenge in quantum computing.

## 12.6 Quantum Threshold Theorems

Perhaps the most groundbreaking discovery in the theory of QECC is the threshold theorem (or rather "theorems" because they must be proved separately for each QECC).

The basic question we are trying to answer is: Are we sure that, by complexifying the quantum hardware with a QECC, we are reducing the overall probability of having an error on a logical qubit, instead of increasing it through the added qubits and gates where errors can occur?

The answer turns out to be yes, provided the error rate $p$ on a single physical qubit or elementary gate is smaller than some threshold value $p_{\text{th}}$.

We can provide an intuitive account of the proof by the following remarks. Suppose the physical error rate is $p$. This is the rate at which $k$-qubit errors occur in a $[[n, k, d]]$ QECC. Then the rate of a $k$-qubit error on an encoded logical qubit is $O(p^e)$. We have already had a glimpse of this on the simple

3-qubit Shor's code to correct $X$-errors. If two $X$-errors occur on physical qubits, then the majority criterion will detect a bit flip in the opposite direction. The correction will then flip the only qubit unaffected by error and the result will be a logical $\bar{X}$ on the encoded logical qubit:

$$\bar{0} = |000\rangle, \quad \bar{1} = |111\rangle \tag{12.13}$$

$$|\psi\rangle = \alpha|000\rangle + \beta|111\rangle \tag{12.14}$$

After two errors on, e.g., bits 1 and 3:

$$|\psi'\rangle = \alpha|101\rangle + \beta|010\rangle \tag{12.15}$$

and the correction will lead to:

$$|\psi''\rangle = \alpha|111\rangle + \beta|000\rangle = \bar{X}|\psi\rangle. \tag{12.16}$$

A similar argument holds for codes correcting $k$-errors, although it may be a bit more difficult to prove it rigorously.

Two errors occur with probability $p^2$ if one error occurs with probability $p$. However, there are several ways in which two errors can occur, and even those when counting also the error correction part of the circuit. So, for a given QECC, the probability of one error (of the same kind of those that are correctable on physical qubits) on a logical qubit is ...

$$p_1 = Cp^2 \tag{12.17}$$

Similar considerations apply to 2-qubit logical gates that are encoded transversally.

Now we understand that the overall result of the QECC protocol is to improve the error rate, if $p < p_1$. This condition depends on the constant $C$.

For typical QECC, $C$ is usually large because there are several ways in which two errors can occur. Also, $C$ is not so obvious to compute and it is known only approximately for most codes.

For Steane's 7-qubit code for example, a lower bound is $C \simeq 10^4$, but it is believed to be in the range of $10^5 - 10^6$.

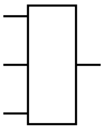
So, a necessary condition for a QECC to be useful is that $p_1 < p$, i.e.,

$$Cp < 1 \quad \text{or} \quad p < \frac{1}{C} = p_{\text{th}}. \tag{12.18}$$

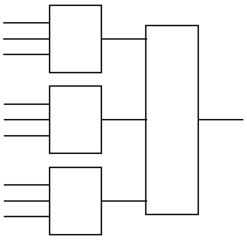
Once this condition is achieved, we can apply a very intuitive and very elegant trick. We now have an encoding for (logical) qubits, with error rate $p_1 < p$.

We know how to implement universal gates on three qubits. Let us then pretend that they are the physical qubits and let us build a QECC (the same as before, but not necessarily) with them.

For example, if the 3-qubit code for correcting $X$ is represented as:



Then our "double" QECC will look like:



This code corrects $X$-errors on the logical qubits encoded at the 1st level, which in turn occur when two $X$-errors occur at the zero-th (physical) level.

Now the rate of $X$-errors on the doubly encoded qubit is:

$$p_2 = Cp_1^2 = C^3p^4 \tag{12.19}$$

We see that the condition $p_2 < p_1$ gives:

$$C^3p^4 < Cp^2 \implies Cp^2 < 1 \implies Cp < 1 \implies p < \frac{1}{C} = p_{\text{th}}. \tag{12.20}$$

We see that this $2^{\text{nd}}$ layer still improves the error rate by the same factor, provided the physical error rate $p$ is below threshold.

More generally, if we concatenate $k$ QECC layers, the error rate will be:

$$p_n = \frac{(Cp)^{2^k}}{C}. \tag{12.21}$$

which can become arbitrarily small by increasing $k$, provided $Cp < 1$.

More precisely, if we require an accuracy $\epsilon$ in the algorithm we wish to run, and that the algorithm is made by a circuit with $p(n)$ gates/locations (where $n$ is the size of the problem and $p(n)$ is a polynomial function). All quantum algorithms are of this kind, that is, with a poly($n$) number of gates. Roughly, each gate/location must be accurate to $\epsilon/p(n)$. Then for a QECC protocol to succeed we need:

$$\frac{(Cp)^{2^k}}{C} \leq \frac{\epsilon}{p(n)}. \tag{12.22}$$

and we may choose $k$ so that the cond. is fulfilled.

<u>*Note:*</u> A ”location” is a point in the circuit where a qubit is left idle or without a gate acting on it at that particular time slice.

Now we can single out a condition on $k$:

$$(Cp)^{2^k} \leq \frac{C\epsilon}{p(n)}. \tag{12.23}$$

$$2^k \log(Cp) = \log\left(\frac{C\epsilon}{p(n)}\right). \tag{12.24}$$

$$2^k = \frac{\log\left(\frac{C\epsilon}{p(n)}\right)}{\log(Cp)}. \tag{12.25}$$

Now, for our strategy to work, the size of the quantum hardware must not grow exponentially with $k$. If one level of QECC requires $d$ gates/locations, then the whole multiple QECC protocol will require $d^k$ gates/locations ($d$ is the factor that multiplies the number of gates/locations of the naked circuit).

But from the previous expression for $2^k$,

$$d^k = \left(\frac{\log\left(\frac{p(n)}{C\epsilon}\right)}{\log\left(\frac{1}{Cp}\right)}\right)^{\log d/\log 2}. \tag{12.26}$$

$$d^k = O\left(\text{poly}\left(\log\left(\frac{p(n)}{\epsilon}\right)\right)\right). \tag{12.27}$$

Then, since the original circuit contains $p(n)$ gates, the total number of gates/locations needed is:

$$O\left(\text{poly}\left(\log\left(\frac{p(n)}{\epsilon}\right)\right)p(n)\right). \tag{12.28}$$

which is only poly-logarithmically larger than the original circuit.

This finally constitutes the proof of the threshold theorem for quantum computation.

*Theorem:* A quantum circuit containing $p(n)$ gates can be executed with probability of error at most $\epsilon$, using:

$$O\left(\text{poly}\left(\log\left(\frac{p(n)}{\epsilon}\right)\right)p(n)\right) \tag{12.29}$$

gates on hardware with physical error rate $p$, provided $p < p_{\text{th}}$, where $p_{\text{th}}$ is determined by the specific QECC protocol being used.

The reason why this theorem is a breakthrough is that it essentially proves that quantum computation is possible, and not forbidden by some fundamental physical limit on quantum hardware.

Today's best QECC, like the surface code, have estimated $p_{\text{th}} = 10^{-2}$. Assuming a physical error rate of $p = 10^{-3}$ for depolarizing errors, this QECC protocol would require $\sim 10^3 - 10^4$ physical qubits per logical data qubit. (This estimate is based on the requirement that the final error rate be $\sim 10^{-10}$, similar to classical circuits).

An improvement of $p$ would make the corresponding number of physical qubits $N$ smaller. Fault tolerance is still far away as the best quantum hardware today can achieve $p \sim 0.01$. An alternative promising field is that of non fault-tolerant quantum computing, using NISQ hardware and hybrid quantum/classical algorithms featuring the use of ”shallow” quantum subroutines.

# Chapter 13

# The Variational Quantum Eigensolver

## 13.1 Variational Quantum Algorithms (VQA)

Variational Quantum Algorithms (VQAs) are a class of hybrid quantum-classical algorithms designed to leverage the computational advantages of quantum devices while utilizing classical resources for optimization. The core idea is to employ a parameterized quantum circuit to perform tasks that are computationally expensive for classical computers. The quantum device prepares a trial quantum state using a set of tunable parameters, and the classical optimizer updates these parameters iteratively based on measurements performed on the quantum device.

Figure 13.1 illustrates the hybrid feedback loop between the quantum device and the classical optimizer.

Figure 13.1: Schematic representation of the VQA workflow. The quantum device prepares a parameterized state and measures the expectation value of the objective function. A classical optimizer updates the parameters iteratively to minimize the objective function.

VQAs are particularly well-suited for the current era of Noisy Intermediate-Scale Quantum (NISQ) devices. Indeed, the iterative nature of VQAs and their reliance on short-depth quantum circuits make them less sensitive to noise compared to algorithms requiring deep quantum circuits. Moreover, VQAs leverage quantum devices for tasks such as state preparation and expectation value estimation. Estimating an expectation value on a NISQ device can be done approximatively. There are two contributions to the error. The first is the statistical error due to the intrinsic probabilistic nature of the-measurement in quantm mechanics – sometimes called *shot noise* – together with the finite number of measurements that are used to estimate the statistical average. The second is an unavoidable finite uncertainty and a systematic bias, both due to the noise in the quantum device. The shot-noise error can be reduced by increasing the number of measurements, while the NISQ error can only be mitigated with a range of techniques called *error mitigation*.

In most VQAs, a parametrized quantum circuit $\mathcal{U}(\boldsymbol{\theta})$ is used to prepare a quantum state $|\Psi(\boldsymbol{\theta})\rangle = \mathcal{U}(\boldsymbol{\theta})|0\rangle$ that depends on a set of variational parameters $\boldsymbol{\theta}$. The quantum state is then measured to estimate the expectation value $\mathcal{L}(\boldsymbol{\theta}) = \langle\Psi(\boldsymbol{\theta})|\, O\, |\Psi(\boldsymbol{\theta})\rangle$ of an observable $O$. The classical optimizer updates the parameters $\boldsymbol{\theta}$ to minimize the objective function, leading to an approximate solution to the problem at hand. The shot-noise error originates from the fact that, in estimating the expectation value, we are using a finite number of measurements to approximate the true expectation value, i.e.

$$\mathcal{L}(\boldsymbol{\theta}) \approx \frac{1}{N}\sum_{i=1}^{N}\langle\Psi(\boldsymbol{\theta})||x_i\,\rangle\langle\, x_i|O|\Psi(\boldsymbol{\theta})\rangle \tag{13.1}$$

$$= \frac{1}{N}\sum_{i=1}^{N}|\langle x_i|\Psi(\boldsymbol{\theta})\rangle|^2\frac{\langle x_i|O|\Psi(\boldsymbol{\theta})\rangle}{\langle x_i|\Psi(\boldsymbol{\theta})\rangle}, \tag{13.2}$$

where the last line highlights that the estimate is a statistical average over the outcomes $x_i$ of the measurements in the computational basis. The shot-noise error is due to the intrinsic probabilistic nature of quantum measurements. To estimate the expectation value with a given accuracy $\epsilon$, the number of measurements $N$ must scale as $N \sim 1/\epsilon^2$. This fact is simply a consequence of the central limit theorem, which states that the error in the average of a random variable decreases as the square root of the number of samples.

In a variational approach, the parametrized state $|\Psi(\boldsymbol{\theta})\rangle$ is expected to be an approximation of the true target state $|\Psi_{\text{target}}\rangle$. Hence, another contribution to the accuracy of the VQA is the approximation error. An upper bound to this error is the distance between the approximate state and the target state, i.e., $|||\Psi(\boldsymbol{\theta})\rangle - |\Psi_{\text{target}}\rangle||$. The approximation error can be reduced by increasing the expressivity of the variational ansatz, i.e., by increasing the number of parameters or the depth of the quantum circuit. However, increasing the expressivity of the ansatz also increases the number of parameters that need to be optimized, which can lead to optimization challenges such as local minima and slow convergence.

Variational Quantum Algorithms have demonstrated significant potential in several domains, including quantum chemistry, optimization, and machine learning. In what follows, we will explore one of the most prominent VQAs, the Variational Quantum Eigensolver (VQE), in detail.

## 13.2 The Variational Quantum Eigensolver (VQE)

The Variational Quantum Eigensolver (VQE) was introduced in 2014 by Peruzzo et al. [? ] as a method to compute the ground state energy of quantum systems. It builds on the principles of the Variational Monte Carlo (VMC) method, a classical algorithm for solving optimization problems in quantum mechanics.

The goal of the VQE algorithm is to find the ground state energy of a given Hamiltonian $\hat{H}$. This is achieved by leveraging the variational principle, which states that for any trial state $|\Psi(\boldsymbol{\theta})\rangle$, parameterized by a set of variational parameters $\boldsymbol{\theta}$, the expectation value of the Hamiltonian provides an upper bound to the ground state energy:

$$E(\boldsymbol{\theta}) = \langle\Psi(\boldsymbol{\theta})|\hat{H}|\Psi(\boldsymbol{\theta})\rangle \geq E_0, \tag{13.3}$$

where $E_0$ is the true ground state energy of $\hat{H}$. By iteratively optimizing the parameters $\boldsymbol{\theta}$, the trial state $|\Psi(\boldsymbol{\theta})\rangle$ is improved until it approximates the ground state, and the energy $E(\boldsymbol{\theta})$ approaches $E_0$.

The variational ansatz $|\Psi(\boldsymbol{\theta})\rangle$ is key to the predictive power of the algorithm. It is constructed using a series of quantum gates that depend on the parameters $\boldsymbol{\theta}$. The choice of ansatz is critical to the success of VQE, as it must balance two competing requirements:

- **Expressivity:** The ansatz must be expressive enough to capture the ground state of the Hamiltonian.

- **Efficiency:** The ansatz must be implementable on a quantum device with a polynomial number of operations.

Today there is a large variety of criteriafor the choice of ansatz. Two broad classes of Ansätze are the *hardware-efficient ansätze*, which are tailored to the native gate set and connectivity of the quantum hardware, and the *physically-motivated ansätze*, which incorporate domain-specific knowledge about the problem. The advantage of hardware-efficient ansaätze is the fact that they can be kept shallow, resulting in a limited gate count, thus reducing the impact of noise. The drawback is that they might not be expressive enough to capture the ground state of the Hamiltonian wiht sufficient accuracy. In addition, the optimization landscape of hardware-efficient ansätze can be more challenging to explore, as they do not account for the actual physics of the problem. Physically-motivated ansätze, on the other hand, are designed to capture the physics of the problem, making them more expressive. However, they can be more challenging to implement on current quantum hardware due to their depth and gate count. Here we won't discuss in detail physically-motivated ansätze, but we will focus on hardware-efficient ansätze.

An example of a hardware-efficient ansatz is illustrated in the figure below. It consists of alternating layers of single-qubit rotations and entangling gates. It is usually designed to be compatible with the gate set and connectivity of specific devices. In the circuit one can identify repeating layers. On layer consists of single-qubit rotations, which are parametrized by the variational parameters $\boldsymbol{\theta}$, and entangling gates, which create entanglement between the qubits. The entanglement is crucial for the ansatz to capture the correlations present in the ground state of the Hamiltonian. The layers are repeated multiple times, each time with different variational parameters, to increase the expressivity of the Ansatz. The depth of the ansatz, i.e., the number of layers, is a hyperparameter that can be tuned to balance expressivity and efficiency. In the circuit below, as ean example, entangling gates only act on nearest-neighbor qubits. Hence, a single layer can only create entanglement between adjacent qubits. By repeating the layers, entanglement can be spread across the entire system. If the ground state of the Hamiltonian is expected to have long-range entanglement on the whole scale of the system size – typically called volume-law entanglement – then an ansatz with only nearest-neighbor entanglement in a single layer, should be repeated at least a number of times of the order of the system size, to capture the long-range entanglement. Alternatively, one can use ansätze that create long-range entanglement in a single layer, but this would come at the cost of increased gate count and depth.

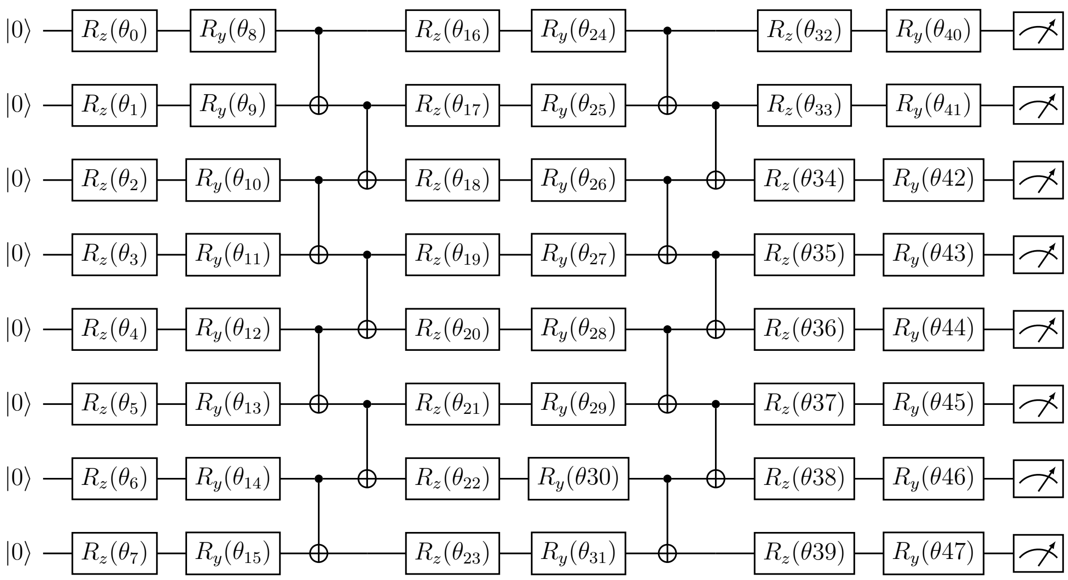

$|0\rangle$ $R_z(\theta_0)$ $R_y(\theta_8)$ $R_z(\theta_{16})$ $R_y(\theta_{24})$ $R_z(\theta_{32})$ $R_y(\theta_{40})$
$|0\rangle$ $R_z(\theta_1)$ $R_y(\theta_9)$ $R_z(\theta_{17})$ $R_y(\theta_{25})$ $R_z(\theta_{33})$ $R_y(\theta_{41})$
$|0\rangle$ $R_z(\theta_2)$ $R_y(\theta_{10})$ $R_z(\theta_{18})$ $R_y(\theta_{26})$ $R_z(\theta 34)$ $R_y(\theta 42)$
$|0\rangle$ $R_z(\theta_3)$ $R_y(\theta_{11})$ $R_z(\theta_{19})$ $R_y(\theta_{27})$ $R_z(\theta 35)$ $R_y(\theta 43)$
$|0\rangle$ $R_z(\theta_4)$ $R_y(\theta_{12})$ $R_z(\theta_{20})$ $R_y(\theta_{28})$ $R_z(\theta 36)$ $R_y(\theta 44)$
$|0\rangle$ $R_z(\theta_5)$ $R_y(\theta_{13})$ $R_z(\theta_{21})$ $R_y(\theta_{29})$ $R_z(\theta 37)$ $R_y(\theta 45)$
$|0\rangle$ $R_z(\theta_6)$ $R_y(\theta_{14})$ $R_z(\theta_{22})$ $R_y(\theta 30)$ $R_z(\theta 38)$ $R_y(\theta 46)$
$|0\rangle$ $R_z(\theta_7)$ $R_y(\theta_{15})$ $R_z(\theta_{23})$ $R_y(\theta_{31})$ $R_z(\theta 39)$ $R_y(\theta 47)$

Figure 13.2: Example of a hardware-efficient ansatz for the VQE algorithm. The ansatz consists of alternating layers of single-qubit rotations and entangling gates.

### 13.2.1 Energy Estimation

The energy $E(\boldsymbol{\theta})$ is estimated by measuring the expectation value of the Hamiltonian $\hat{H}$. We assume here that the Hamiltonian can be expressed as a sum of Pauli operators, i.e.,

$$\hat{H} = \sum_{i}^{M} h_i \hat{P}_i, \tag{13.4}$$

where $\hat{P}_i$ are tensor products of Pauli matrices, and $h_i$ are coefficients. The idea underlying this assumption is that any Hermitian operator can be expressed as a linear combination of Pauli operators, although in general the number of terms in the sum scales exponentially with the number of qubits. Once the Pauli-sum expression has been obtained, the energy can be computed as:

$$E(\boldsymbol{\theta}) = \sum_{i}^{M} h_i \langle \Psi(\boldsymbol{\theta}) | \hat{P}_i | \Psi(\boldsymbol{\theta}) \rangle. \tag{13.5}$$

This requires measuring the expectation value of each Pauli operator $\hat{P}_i$ on the quantum device. The expectation value is estimated by performing measurements in the appropriate basis and averaging the results over multiple shots to reduce statistical error. Notice that, if the number of measurements required to reach a given statistical accuracy is $N$, then the total number of measurements required to estimate the energy is $N$ times the number of terms $M$ in the Pauli-sum expression of the Hamiltonian, i.e., $N_{meas} = NM$. This can be a significant computational overhead, especially for large systems.

The main application of VQE is in quantum chemistry, where the Hamiltonian is typically expressed in terms of fermionic operators describing electrons occupying molecular orbitals. There is a textbook procedure to map fermionic operators to qusums of Pauli operators, which is based on the Jordan-Wigner or Bravyi-Kitaev transformations. The resulting Hamiltonian however typically has a large number of terms, which can approach millions for molecules of practical interest. The number of terms can be reduced by various techniques. An example is to group together terms that commute with each other, and measure them simultaneously. This is a complex combinatorial problem and sometimes the most advanced methods in approximately solving combinatorial problems have been advocated to lower the computational overhead of VQE in quantum chamistry. However, these techniques do not lead in general to a dramatic reduction of the number of terms that must be measured independently, and the search for a randically different variational approach to quantum chemistry is still ongoing.

### 13.2.2 The Optimization Process

The primary objective of the Variational Quantum Eigensolver (VQE) is to minimize the energy $E(\boldsymbol{\theta})$ of the trial state $|\Psi(\boldsymbol{\theta})\rangle$, as defined by:

$$E(\boldsymbol{\theta}) = \langle\Psi(\boldsymbol{\theta})|\hat{H}|\Psi(\boldsymbol{\theta})\rangle. \tag{13.6}$$

This optimization process is iterative and involves updating the variational parameters $\boldsymbol{\theta}$ based on the results of measurements performed on the quantum device. The workflow is as follows:

1. **Initialize the Parameters:** Begin with an initial guess for $\boldsymbol{\theta}$.
2. **Measure the Energy:** Evaluate $E(\boldsymbol{\theta})$ using the quantum device.
3. **Optimize the Parameters:** Update $\boldsymbol{\theta}$ using a classical optimization algorithm to reduce $E(\boldsymbol{\theta})$.
4. **Iterate Until Convergence:** Repeat steps 2 and 3 until the energy converges to a minimum.

There are two broad strategies to optimize the loss function: gradient-based optimization and gradient-less optimization.

In gradient-based optimization, the parameters $\boldsymbol{\theta}$ are updated iteratively in the direction of the negative gradient of $E(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta\nabla E(\boldsymbol{\theta}^{(k)}), \tag{13.7}$$

where $\eta > 0$ is the learning rate, and $\nabla E(\boldsymbol{\theta})$ is the gradient of the energy with respect to the parameters. These approaches require estimating the gradient of the energy, which can be challenging on quantum devices. Indeed, for most variational ansätze, the gradient of the energy is not known analytically, and must be estimated using finite differences or other techniques. This implies that not only one has to carry out a large number of measurements to estimate the energy, but in principle also a much larger number of measurements to estimate each component of the gradient. This can be a significant computational overhead, especially for large systems. There are some techniques that can be used to reduce the number of measurements required to estimate the gradient, such as the stochastic gradient descent, which estimates only the derivative with respect to a random subset of parameters at each gradient-descent step. However, these techniques can introduce additional noise in the gradient estimation, which can affect the convergence of the optimization process.

In hardware-efficient ansätze like the one illustrated above, the dependence on the angle $\theta$ of each gate is of a specific form involving trigonometric functions. This allows for the use of the parameter-shift rule to estimate the gradient of the energy. The parameter-shift rule is a technique that removes the additional error introduced by finite differences in the gradient estimation. The exact derivative of the energy with respect to a parameter $\theta$ is given by:

$$\frac{\partial E}{\partial \theta} = \frac{1}{2}\left[E(\theta + \frac{\pi}{2}) - E(\theta - \frac{\pi}{2})\right]. \tag{13.8}$$

Notice that the parameter shift is not a small quantity, like it would be the case in the usual finite difference calculation. This is due to the specific dependence of the ansatz on trigonometric functions and eliminates the error due to the finite difference approximation.

The drawbacks of gradient-based optimization are mostly of three kinds. First, the possibility to get stuck in local minima of the landscape of the loss function. This is partially avoided using advanced gradient-based techniques such as ADAM, which make use of the past history of the minimization path in parameter space, and in particular of the momentum of the motion along the landscape of the loss function. The second drawback is obviously the significant overhead in the estimation of the gradient. The third, and arguably the most important drawback, is the occurrence of *barren plateaux*. Barren plateaux are regions of the parameter space where the gradient of the loss function is very small, and the optimization process becomes very slow. Barren plateaux are a well-known issue in the optimization of quantum circuits, and they become more pronounced as the number of qubits increases. Some argue that barren plateaux are a universal feature of the variational approach, as they would arise from the intrinsically exponential complexity of computing a quantum state of a many-body system. Although this is a very interesting hypothesis, it is not yet clear whether it is the case. An empirical observation is that barren plateaux are much less pronounced in physically-motivated ansätze, which are designed to capture the physics of the problem, and therefore have a landscape of the loss function which is more adapted to the physically relevant regions of the Hilbert space. Finally, the variational approach is also severely affected by noise, which can flatten the landscape of the loss function, thus making gradient-based methods less effective, and can introduce additional biases by actually modifying the location of the minimum of the loss function.

Let us just mention an approach called *Quantum Natural Gradient*, which is a modification of the gradient descent update rule that incorporates the Quantum Geometric Tensor (QGT), which provides information about the curvature of the parameter space. The update rule becomes:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta F^{-1} \nabla E(\boldsymbol{\theta}^{(k)}), \tag{13.9}$$

where $F$ is the QGT, capturing the geometry of the quantum state manifold. Here we won't discuss the details of this approach, and refer the interested reader to the literature.

Gradient-free methods provide alternatives to gradient-based techniques, especially when gradients are difficult or costly to compute. These methods rely on direct evaluation of the energy at many points in the parameter space to heuristically determine the optimal parameters. More precisely, at each iteration these methods evaluate the energy at many points and use a heuristic criterion to determine a new set of points for the next iteration, which are more likely to be close to the actual minimum of the loss function. Among the many gradient-free optimization methods, we can mention the Nelder-Mead optimization, sometimes called the simplex method or amoeba method, and evolutionary algorithms such as the genetic algorithm or the particle swarm optimization. Another, virtually gradient-free optimization method for VQE, is the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm. SPSA is a robust and efficient method that requires only two energy evaluations per iteration, making it suitable for noisy quantum devices. The SPSA algorithm is based on the simultaneous perturbation of all parameters, which allows for a more efficient estimation of the gradient. The algorithm has been successfully applied to various quantum optimization problems, including the VQE algorithm.

which is a direct search method that does not require the computation of the gradient, and Bayesian optimization, which uses a probabilistic model to explore the parameter space. The most popular gradient-free optimization method for VQE is the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm. SPSA is a robust and efficient method that requires only two energy evaluations per iteration, making it suitable for noisy quantum devices. The SPSA algorithm is based on the simultaneous perturbation of all parameters with a random small displacement of each parameter. One can show that this approach provides an almost unbiased estimate of the gradient of the loss function. More precisely, the SPSA algorithm estimates the gradient of the loss function as:

$$\frac{\partial E}{\partial \theta_j} \approx \frac{E(\boldsymbol{\theta} + \Delta) - E(\boldsymbol{\theta} - \Delta)}{2\Delta_j}, \tag{13.10}$$

where $\Delta$ is a random vector of random perturbations.

It is not within the scope of the present discussion to provide a detailed comparison of the various optimization methods. The reader is encouraged to consult the literature for a more in-depth analysis of the advantages and limitations of each approach. However, it is important to note that the choice of optimization method can significantly impact the convergence and performance of the VQE algorithm. The choice of optimization method should be guided by the specific characteristics of the problem at hand, such as the landscape of the loss function, the presence of noise, and the availability of computational resources.

A considerable number of review articles exist on the VQE algorithm and its applications. A non exhaustive list includes [4, 5, 6, 7, 8, 9, 10].

# Chapter 14

# The Quantum Approximate Optimization Algorithm

## 14.1 Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA) was introduced in 2014 as a VQA to solve combinatorial optimization problems with a variational approach on a NISQ quantum computer.

Combinatorial optimization problems (COPs) are ubiquitous, and virtually any kind of optimization problem in math, phys, eng, etc., can be reduced to one of these problems.

A general class of COP is the *boolean satisfiability problem*, or SAT, whereby if $\{x_j\}$ are boolean variables and $R(x_1, x_2, \ldots, x_n)$ a boolean formula, the problem consists in determining if there exists a set of values TRUE/FALSE for $x_1, x_2, \ldots, x_n$ such that $R =$ TRUE.

It is intuitive to understand how SAT enables the resolution of very many optimization problems. This is also formally established by the Cook-Levin theorem, which was the 1st proof of NP-completeness of a computational problem, proving that SAT is NP-complete.

As an example, a particular instance of SAT is the 3-SAT problem, where

$$R = R_1 \wedge R_2 \wedge \cdots \wedge R_m, \tag{14.1}$$

$$R_j = y_{j_1} \vee y_{j_2} \vee y_{j_3}, \quad y_{j_k} \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}. \tag{14.2}$$

Related to 3-SAT, and to its generalization, the $k$-SAT problem, is the MAX-SAT problem, where one tries to *find the solution that maximizes* the number of $R_j =$ TRUE. This is more of an optimization problem than a satisfiability problem and is NP-Hard. The corresponding decision problem is NP-complete.

Another example of COP is the *Traveling Salesman Problem*, in which you have to find the optimal route that touches on all cities in a map. The problem is NP-Hard and the corresponding decision problem, in which you have to decide whether given a length $\ell$, the graph has a solution with a route of length at least $\ell$, is NP-complete. Another important example is the MAX-CUT problem, which we will introduce in a moment.

An important feature of all these problems is the fact that they can be mapped, more or less efficiently, onto a classical Hamiltonian of interacting spins, often called an Ising spin-glass (ISG) Hamiltonian[1]. An ISG Hamiltonian has the form

$$H(s_1, s_2, \ldots, s_N) = -\sum_{i<j} J_{ij} s_i s_j - \sum_{i=1}^{N} h_i s_i, \tag{14.3}$$

where $s_j = \pm 1$. The task of finding the classical ground state, that is, the configuration of the $s_j$'s that minimizes $H$, is known to be NP-Hard.

A very important development in the solution of these problems comes with the idea of *Quantum Adiabatic Optimization* (QAO)[2].

The idea of QAO is the following. Consider $H$ not as a classical but as a quantum Hamiltonian:

$$H_p = -\sum_{i<j} J_{ij} Z_i Z_j - \sum_{i=1}^{N} h_j Z_j. \tag{14.4}$$

This represents the exact same problem, because $H_p$ is diagonal (*thus classical*) in the basis of eigenstates of $Z_j$. We then introduce another Hamiltonian

$$H_0 = -h_0 \sum_{i=1}^{N} X_j. \tag{14.5}$$

This Hamiltonian is separable and therefore simple to solve. Each $X_j$ has eigenstates $|+\rangle$ and $|-\rangle$, so that the lowest energy state is $|+\rangle^{\otimes N}$.

Now define the *time-dependent Hamiltonian*:

$$H(t) = \left(1 - \frac{t}{T}\right) H_0 + \frac{t}{T} H_c. \tag{14.6}$$

At time $t = 0$, $H(t) = H_0$, and at $t = T$, $H(T) = H_c$.

The idea of QAO is then to prepare a quantum system in the state $|+\rangle^{\otimes N}$ at time $t = 0$, set $T$ large enough so that the dynamics is very slow, and let the system evolve until $t = T$. If $T$ is large enough, the system *adiabatically* stays in the ground state at all $t$ and ends up in the ground state of $H_p$ which we are searching.

The choice of $H_0$ ensures that $|+\rangle^{\otimes N}$ has equal components on all states of the comp. basis, so that QAO always works. It is also why we need quantum mechanics. The idea is reminiscent of quantum parallelism. We start the annealing with a superposition of all separable states. Then the quantum annealing protocol explores in parallel all solutions of the classical problem.

[1] See `arXiv:1302.5843` for a review.

[2] See `arXiv:0801.2193` for a review.

## 14.2 QAOA on a digital quantum computer

QAO can be achieved by digital quantum computing. The time evolution associated to $H(t)$ is

$$U(t,0) = \mathcal{T} \exp \left( \int_0^t H(t')dt' \right). \tag{14.7}$$

Do not worry much about this expression. It is just the generalization of the time-evolution operator to the case in which the Hamiltonian is time-dependent. Here we will introduce a simplification consisting in approximating the Hamiltonian $H(t)$ as *piecewise constant*, i.e. time-independent on each short time interval $\Delta t$. Notice that $[H_0, H_c] \neq 0$. We can divide the interval $T$ into $M$ steps $\Delta t = \frac{T}{M}$, then

$$U(T,0) = U(T, T-\Delta t)U(T-\Delta t, T-2\Delta t)\dots U(\Delta t, 0). \tag{14.8}$$

Now, if $\Delta t$ is small, i.e., $\langle H \rangle \Delta t \ll 1$, the piecewise-constant approximation allows to rewrite each time-evolution operator over an interval $\Delta t$ using the expression we have introduced for time-independent Hamiltonians

$$U(T-k\Delta t, T-(k-1)\Delta t) \approx e^{-iH(k\Delta t)\Delta t}. \tag{14.9}$$

Let us also call $H_0(t) = (1-\frac{t}{T})H_0$ and $H_c(t) = \frac{t}{T}H_c$. The Trotter formula tells us that:

$$e^{-iH(k\Delta t)\Delta t} \approx \left( e^{-iH_0(k\Delta t)\Delta t/M} e^{-iH_c(k\Delta t)\Delta t/M} \right)^M. \tag{14.10}$$

where the Trotter error is $\mathcal{O}\left(\frac{(\Delta t)^2}{M}\right)$. Both $e^{-iH_0 t}$ and $e^{-iH_c t}$ can be computed analytically as a product of Pauli operators, and therefore translated directly into a quantum circuit. If we execute the quantum circuit, at the output we will have (an approximation of) the ground state of $H_c$, which is a state of the computational basis. One can then simply read out the classical solution by measuring each qubit in the computational basis.

This seems to solve the problem! Where is the trick?

The issue hides in the *adiabatic theorem*, which states that the distance between the actual state $|\psi(T)\rangle$ at the end of the evolution and the actual solution $|\psi_0\rangle$ is bounded by

$$\| |\psi(T)\rangle - |\psi_0\rangle \| \lesssim \frac{\langle H \rangle}{T\Delta^2}, \tag{14.11}$$

where $\Delta$ is the *minimal gap*, defined as $\Delta = E_1 - E_0$, where $E_0$ is the ground state energy of $H$, and $E_1$ is the next level energy. For systems of increasing size, $\Delta$ can decrease exponentially to zero, thus requiring a time window $T$ very long for *adiabatic* evolution.

Adiabatic means without transfer of energy. Any time-dependent term in a Hamiltonian implies an exchange of energy. This is why one never gets to the true ground state $|\psi_0\rangle$, but to a state that has components on excited states $|\psi_1\rangle$, etc. The transfer of energy in this case is called the *Landau-Zener transition.*

One can show that the probability of having the ground state at $T$, $P_{\mathrm{GS}}(T)$, obeys the Landau-Zener formula

$$\log(1 - P_{\mathrm{GS}}(T)) \propto -T\Delta^2. \tag{14.12}$$

There were several attempts at solving this issue. It turns out that a better strategy consists in a *diabatic* time evolution. In simple terms, one adopts a short time $T$, but chooses a non-uniform time evolution:

$$H(t) = (1 - f(t))H_0 + f(t)H_C \tag{14.13}$$

with the conditions $f(0) = 0$ and $f(T) = 1$.

In practice, one can parametrize the function $f(t)$ with several parameters. Then, for a given $T$, one runs the diabatic time evolution several times, by adjusting the parameters each time (namely varying $f(t)$), to search for the minimum of the expectation value of the energy $\mathcal{E} = \langle \Psi(T)|H_c|\Psi(T)\rangle$.

This is exactly the idea at the basis of QAOA. It turns out that, by optimizing $f(t)$, one can achieve a *dramatic improvement* in the time $T$ needed to achieve good convergence.

In the QAOA algorithm one has set of real-valued parameters

$$\vec{\xi} = (\vec{\gamma}, \vec{\beta}) = (\gamma_1, \ldots, \gamma_p, \beta_1, \ldots, \beta_p). \tag{14.14}$$

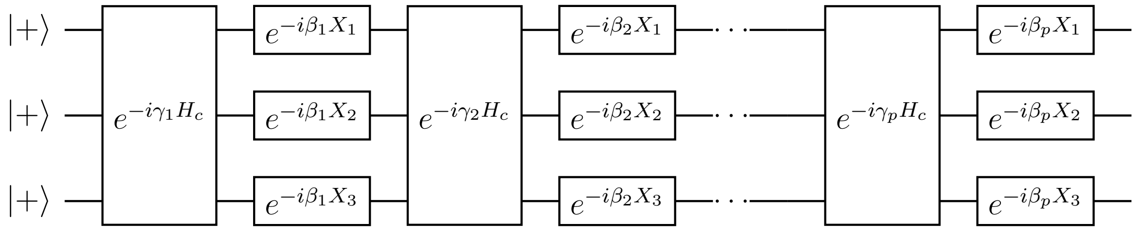


Figure 14.1: The quantum circuit of the QAOA algorithm

The quantum circuit is designed as follows

- The initial state is prepared in $|+\rangle^{\otimes N}$.
- The operators $e^{-i\gamma H_c}$ and $e^{-i\beta H_0}$ are applied alternately $p$ times, with $j = 1, \ldots, p$.
- The final state is measured to estimate $\mathcal{E}(\vec{\xi}) = \langle H_c \rangle$. $H_c$ is a sum of Pauli operators, so it is straightforward to estimate.

Then $(\vec{\gamma}, \vec{\beta})$ are varied following a variational scheme. We can choose between gradient-based methods, where the estimate of the gradient of the loss function instructs about how to change the parameters, and gradient-less methods, where the minimum is searched using some heuristic exploration of parameter space. The simplest example of gradient-based method is the gradient descent. Assume that at iteration $n$ the parameters are $\vec{\xi}_n$. Then at iteration $n + 1$, the new parameters are computed as

$$\vec{\xi}_{n+1} = \vec{\xi}_n - \eta \nabla \mathcal{E}(\vec{\xi})|_{\vec{\xi}=\vec{\xi}_n} . \tag{14.15}$$

The gradient-based approaches may be very accurate if the starting point in parameter space is reasonably close to the actual minimum. However, they require estimating the gradient, which is not possible analytically and therefore requires to execute a large number of circuits and measurements to estimate each gradient component. Advanced methods include the parameter-shift rule, stochastic gradient descent, or the simultaneous perturbation stochastic approximation. Gradient-less methods instead inlcude the Nelder-Mead or simplex method, and more advanced approaches like e.g. genetic algorithms. There is consensus that the overhead in the number of circuit measurements to estimate the gradient, or to estimate the loss function in many points in parameter space in gradient-less methods, is roughly the same. All these issues are discussed in great detail in a recent review on QAOA.[3]

One can show that this approach can be traced back to the diabatic scheme by setting

$$f\left(t_i = \sum_{j=1}^{i} \left(|\gamma_j| + |\beta_j|\right) - \frac{1}{2}\left(|\gamma_i| + |\beta_i|\right)\right) = \frac{\gamma_i}{|\gamma_i + \beta_i|}, \tag{14.16}$$

with

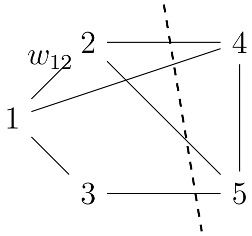
$$T_p = \sum_{i=1}^{p} \left(|\gamma_i| + |\beta_i|\right). \tag{14.17}$$

A key parameter is the number of cycles $p$. Differently from QAO, the QAOA generates a monotonous variation of the quality of the result, which improves with increasing $p$, because of the variational nature of the problem. It is established that, under reasonable assumptions, QAOA cannot be efficiently simulated classically even for $p = 1$.

A final remark concerns the parameter optimization. We know that in the adiabatic limit $T \to \infty$ quantum annealing will lead to the optimal solution independently of the *annealing schedule*, i.e. the particular form of $f(t)$. This property reflects in the QAOA algorithm. It has been demonstrated that, if a sufficiently large number of cycles $p$ is used, then there is an optimal set of parameters $(\vec{\gamma}, \vec{\beta})$ which leads to an approximate optimal solution independently of the instance of the problem. Using this optimal set, one can then avoid the optimization process. The issue is that the number of cycles $p$ required to achieve this optimization-less regime grows with the size of the problem, and current NISQ architectures can't execute very deep circuits because of the excessive noise of current quantum hardware.

## 14.3 Max-Cut

A particular instance of QAOA is the solution of the Max-Cut problem. The Max-Cut problem is defined on a graph as follows. Consider a graph as shown below,



where, $w_{ij}$ are weights and each node can take values ↑ or ↓, $w_{ij} > 0$.[4] An edge contributes with its weight $w_{ij}$ only if nodes $i$ and $j$ are *anti-aligned*. The goal is to find a cut of

[3] K. Blekos et al., Physics Reports 1068, **1** (2024). https://arxiv.org/abs/2306.09198

[4] We show just the edge $w_{12}$ to keep the figure uncluttered.

the graph (like the the dashed line) which maximizes the sum of the weights. As usual, the decision problem *“Is there a cut whose weight is $\geq K$?”* is NP-complete and the corresponding optimization problem, “*Find the best cut*”, is NP-Hard.

The problem is equivalent to maximizing the following Ising Hamiltonian

$$H_c = \sum_{\langle i,j \rangle} w_{ij} \left( \frac{\mathbb{1} - Z_i Z_j}{2} \right). \tag{14.18}$$

If we define the approximate ratio

$$r^* = \frac{\langle H_c \rangle_{\text{approx}}}{\langle H_c \rangle_{\text{max}}}, \tag{14.19}$$

then, even finite values of $r^* < 1$ can be shown to be NP-Hard. Therefore, QAOA provides an exp. speedup and may in the next future bring to another proof of quantum supremacy.

Check out the *Variational Quantum Factoring* and its recent implementation on IBM machines, which reduces the factoring problem to a QAOA instance with a 4-local spin Ising model! (Please refer to arXiv:2012.07825).

# Bibliography

[1] John Preskill. Lecture notes for physics 229: Quantum information and computation. http://theory.caltech.edu/~preskill/ph229/, 1998. California Institute of Technology, Last accessed: September 6, 2024.

[2] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, Cambridge, 10th anniversary edition edition, 2010.

[3] Scott Aaronson. The complexity of quantum states and transformations: From quantum money to black holes, 2016.

[4] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. The Variational Quantum Eigensolver: a review of methods and best practices. *Physics Reports*, 986:1–128, November 2022. arXiv:2111.05176 [quant-ph].

[5] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, September 2021. Number: 9 Publisher: Nature Publishing Group.

[6] Dmitry A. Fedorov, Bo Peng, Niranjan Govind, and Yuri Alexeev. VQE Method: A Short Survey and Recent Developments, August 2021. arXiv:2103.08505 [quant-ph].

[7] Yudong Cao, Jonathan Romero, Jonathan P. Olson, Matthias Degroote, Peter D. Johnson, Mária Kieferová, Ian D. Kivlichan, Tim Menke, Borja Peropadre, Nicolas P. D. Sawaya, Sukin Sim, Libor Veis, and Alán Aspuru-Guzik. Quantum Chemistry in the Age of Quantum Computing. *Chemical Reviews*, 119(19):10856–10915, October 2019. arXiv:1812.09976 [quant-ph].

[8] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qute.201900070.

[9] Han Qi, Sihui Xiao, Zhuo Liu, Changqing Gong, and Abdullah Gani. Variational quantum algorithms: fundamental concepts, applications and challenges. *Quantum Information Processing*, 23(6):224, June 2024.

[10] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92(1):015003, March 2020. Publisher: American Physical Society.