M1 MATHÉMATIQUES DE L'INFORMATION, CRYPTOGRAPHIE

Travail d'Étude et de Recherche

Algorithme de Shor

Xabier LEGASPI JUANATEY

Clément LOUIS

Année 2017-2018

UNIVERSITÉ DE RENNES 1

"In the midst of chaos, there is also opportunity."
—Sun Tzu, The Art of War

Abstract

A *quantum algorithm* is an algorithm that is run with a quantum computer. It exploits the characteristics of quantum states such as the principle of superposition. These lead in certain problems to the dramatic reduction of its complexity. On this report we will show how a quantum algorithm works dealing with an illustrative example: Shor's algorithm for the problem of factoring integers, in which RSA is based.

Résumé

Un *algorithme quantique* est un algorithme qui est executé avec un ordinateur quantique. Il exploite des caractéristiques des états quantiques, par exemple le principe de superposition. Ceux-ci conduisent pour certains problèmes à une réduction drastique de leur complexité. Dans ce report, on montrera le fonctionnement d'un algorithme quantique en travaillant avec un exemple illustratif: l'algorithme de Shor pour le problème de la factorisation des entiers, sur lequel RSA est basé.

Contents

Introduction

Electronic components of nowadays classical computers have been developed thanks to the quantum comprehension of atomic phenomenal. The functioning of the *transistor*, discovered in 1974 by Bardeen, Brattain and Schockley, is based in this comprehension. However, it is also the laws of quantum theory who impose a limitation to this development since quantum effects take place at microscopical scale and there is an increasing ultra-miniaturization of electronic components. This miniaturization follows two laws stated in 1965 by *Gordon Moore*, co-founder of *Intel*:

(1) Every 18 months, the number of transistors per square inch on integrated circuits (chips) is multiplied by a factor of 2.

(2) The capital cost of every new generation of integrated circuits is also multiplied by a factor of 2.

There will come a time when the bit of information will be stocked at an atomic level. And then, quantum effects will become predominant. The electrons will reveal their quantum nature that is, among others, probabilistic. Transistors could no longer be in one of the states 1 or 0 but in a *superposition* of both of them, with some certain probability of being in the state 1 or 0. This kind of behavior leaves us with two options: we must adapt the architecture of computers in order to minimize the impact of quantum effects, or, change radically the architecture embracing these quantum effects. In 1982, the Nobel prize-winning physicist *Richard Feynman* [2] thought up this second idea of a *"quantum computer"*. Quantum information theory follows this approach.

Given that quantum information has many unusual properties, which we will not be covering on these notes, it might have been expected that quantum theory would have a profound impact on our understanding of computation. That this is spectacularly true came as a bolt from the blue unleashed by *Peter Shor* [10],[11] in April, 1994. Shor's algorithm was a monumental discovery not only because it provides exponential speedup over the fastest classical algorithms, but because a number of algorithms for public-key cryptography, including the commonly used RSA algorithm, depend on the fact that there

is no known efficient classical algorithm to factor integers into prime numbers [9]. Shor demonstrated that the realization of a full-scale quantum computer would have the potential to provide a truly significant increase in computing speed, at the same time pointing out the possible implications to the field of cryptography. For both reasons, Shor's discovery sparked a great deal of interest in the design of quantum algorithms and computers that endures today.

In these notes there is no new result or proof. Essentially, we follow the discussions of *F. Xi Lin* [4], *D. Petritis* [7] and *M.A. Nielsen, I.L. Chuang* [6]. We have omitted the proof of some results due to length constraints and we will not talk about the theory of the complexity; we have also included an appendix with some implementations in the *C* language.

Figure 1: Peter Shor.

Notation

– The following sets will be denoted as it follows:

  · $\mathbb{N}$ the set of natural numbers including the zero,

  · $\mathbb{Z}$ the ring of the integers,

  · $\mathbb{Q}$ the field of rational numbers,

  · $\mathbb{R}$ the field of real numbers,

  · $\mathbb{C}$ the field of complex numbers,

  · $\mathbb{Z}/N\mathbb{Z}$ the ring of residues (mod N).

– The intersection of two sets is denoted by $\cap$.
– If $a, b \in \mathbb{Z}$, its greatest common divisor is denoted by $gcd(a, b)$.
– If $a, b \in \mathbb{Z}$, $[\![a, b]\!]$ denotes an integer interval.
– If $a \in \mathbb{N}$, its binary representation is denoted by $\langle a_{n-1} \dots a_0 \rangle_2$.

Chapter 1

Elements of Quantum Theory

This chapter provides all the necessary background knowledge of quantum theory needed for a thorough grasp of quantum algorithms. The only prerequisite for understanding it is some familiarity with elementary linear algebra and probability theory. We have followed the chapter 2 of [8] and the chapter 2 of [7]. For the historical notes we have used the article [1].

1.1 Historical Notes

Quantum mechanics is the theory of the description of matter at microscopical scale. It was developed around the years 1900-1940 beginning as a set of controversial mathematical explanations of experiments that the math of classical mechanics could not explain. Multiple scientists contributed to a foundation of the revolutionary principles that gradually gained acceptance and experimental verification.

In 1900, German physicist *Max Planck* sought to explain the distribution of colors emitted over the spectrum of electromagnetic radiation. When making physical sense of the equation he had derived to describe this distribution, Planck realized it implied that combinations of only certain colors were emitted, specifically those that were whole-number multiples of some base value. Somehow, colors were quantized! This was unexpected because light was understood to act as a wave, meaning that values of color should be a continuous spectrum. What could be forbidding atoms from producing the colors between these whole-number multiples? This seemed so strange that Planck regarded quantization as nothing more than a mathematical trick. Planck's equation also contained a number that would later become very important to future development of quantum mechanics; today, it's known as *"Planck's Constant"*.

In 1905, *Albert Einstein* published a paper, *"Concerning an Heuristic Point of View Toward the Emission and Transformation of Light"*, in which he envisioned light traveling not as a wave, but as some manner of "energy quanta"; as a particle. This packet of energy could *"be absorbed or generated only as a whole"*, specifically when an atom *"jumps"* between quantized vibration rates. This would also apply, as would be shown a few years later, when an electron *"jumps"* between quantized orbits. Under this model, Einstein's *"energy quanta"* contained the energy difference of the jump; when divided by Planck's constant, that energy difference determined the color of light carried by those quanta. With this new way to envision light, Einstein explained how certain colors of light could eject electrons off metal surfaces, a phenomenon known as the *"photoelectric effect"*.

Roughly two decades after Einstein's paper, the term *"photon"* was popularized for describing energy quanta, thanks to the 1923 work of *Arthur Compton*, who showed that light scattered by an electron beam changed in color. This showed that particles of light (photons) were indeed colliding with particles of matter (electrons), thus confirming Einstein's hypothesis. By now, it was clear that light could behave both as a wave and a particle, placing light's "*"wave-particle duality"* into the foundation of quantum mechanics.

In 1924, *de Broglie* used the equations of Einstein's theory of special relativity to show that particles can exhibit wave-like characteristics, and that waves can exhibit particle-like characteristics. Then in 1925, two scientists, working independently and using separate lines of mathematical thinking, applied de Broglie's reasoning to explain how electrons whizzed around in atoms (a phenomenon that was unexplainable using the equations of classical mechanics). In Germany, physicist *Werner Heisenberg* accomplished this by developing *"matrix mechanics"*. Austrian physicist *Erwin Schrödinger* developed a similar theory called *"wave mechanics"*. Schrödinger showed in 1926 that these two approaches were equivalent.

Many formulations of quantum mechanics have being proposed so far. All of them are totally satisfactory from the point of view of computation, predictive power and adaptedness to explain diverse experiments. However, none of the existing them is philosophically and epistemologically satisfactory, because quantum mechanics is still a partial theory. We will work with the most straightforward one, based in the Hilbert Space formalism, introduced by *John von Neumann* [5] in 1932.

1.2 Physical Systems

A theory is a mathematical model of the physical universe.

Definition 1.2.1
A *physical system* or *system* $\mathfrak{S}$ is a portion of the physical universe chosen for analysis. Everything outside the system is known as the *environment*. The environment is ignored except for its effects on the system. When we ignore these effects, we say that the system is *isolated* or *closed*.

Remark 1.2.2
We will only work with isolated systems.

To charaterize a model of a system, we need to specify how it will represent states, observables, measurements and how its dynamics is.

Definition 1.2.3 (I) A *state* is a complete description of a physical system.

(II) An *observable* is a property of a physical system that in principle can be measured.

(III) A *measurement* is a process in which "$information$" about the state of a physical system is acquired by an observer.

(IV) The *dynamics* is the way a physical system evolves over the time.

Suppose we have a system $\mathfrak{S}$ and we are an observer who want to obtain some "$information$" of the system by means of a physical device X that answers our questions about it. We have the following procedure:

(1) The system is prepared in some natural or artificial initial condition: the state ρ.

(2) The system enters in contact with the physical device X designed to do the measurement of a given observable. In other words, X "$asks$" the system the question we have made.

(3) The system returns some information to X in the form of some outcome: the *answer*.

The whole physics relies on the following postulates. Considering them, we will present quantum theory as an extension of the classical probability theory.

Postulate 1.2.4 (*Answers*)
The set of the answers is a measurable space $(\mathbb{X}, \mathcal{X})$.

Postulate 1.2.5 (*Statistical reproductibility of experiments*)
Let $n \in \mathbb{N}$ and let $\mathfrak{S}_1, ..., \mathfrak{S}_n$ be n copies identically prepared of the same system $\mathfrak{S}$.

(I) If we make the same measurement on $\mathfrak{S}_i$ $\forall i$, the outcomes will take random values scattered around some central ones.

(II) As $n \to \infty$, the empirical distribution of the observed data tends to some probability measure.

1.3 Postulates of Classical Systems

Under the formalism of the previous section, classical probability theory is described by the following postulates.

Postulate 1.3.1 (*States*)
In a classical system, a state is a probability measure ρ on a measurable space $(\Omega, \mathcal{F})$.

Postulate 1.3.2 (*Observables*)
In a classical system, an observable is a real valued random distribution $X\colon (\Omega, \mathcal{F}) \to (\mathbb{X}, \mathcal{X})$, $\mathbb{X} \subset \mathbb{R}$.

Postulate 1.3.3 (*Measurement*)
In a classical system, the measurement of an observable X prepares an element of $\mathbb{X}$, and the observer learns the value of the correspondent element. If the classical state just prior to the measurement is ρ, then the outcome x is obtained with *a priori* probability

$$\mathbb{P}_\rho(x) := \rho(X^{-1}(x))$$

Postulate 1.3.4 (*Dynamics*)
The time evolution of a closed classical system is described by an invertible measurable transformation $T\colon (\Omega, \mathcal{F}) \to (\Omega, \mathcal{F})$ leaving the set of states invariant.

Postulate 1.3.5 (*Composite Systems*)

(I) If the measurable space of two classical systems $\mathfrak{S}_1, \mathfrak{S}_2$ are, respect., $(\Omega_1, \mathcal{F}_1), (\Omega_2, \mathcal{F}_2)$, then their composite classical system $\mathfrak{S}_1\mathfrak{S}_2$ has for measurable space $(\Omega_1 \times \Omega_2, \mathcal{P}(\Omega_1 \times \Omega_2))$.

(II) If the systems $\mathfrak{S}_1, \mathfrak{S}_2$ are prepared, respect., in the states ρ_1, ρ_2, then the composite system's state is their conjoint measure of probability ρ.

Definition 1.3.6
A closed system described by all the above postulates is a *classical system*.

1.4 Elements of Hilbert Spaces

We introduce or recall some notions of the *Hilbert space*, the essential tool that permits us to study quantum systems theoretically.

Definition 1.4.1
A *Hilbert space* is a $\mathbb{C}$-vector space $\mathcal{H}$ together with a inner product $\langle\cdot|\cdot\rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{C}$ that is also a complete normed space with respect to the induced norm $\|\cdot\| = \langle\cdot|\cdot\rangle^{\frac{1}{2}}$.

Notation 1.4.2 (*Dirac's Bra and Ket Notation*)
In quantum physics, vectors of a Hilbert space $\mathcal{H}$ are called "*kets*" and are denoted by the symbol $|\cdot\rangle$. If we want to specify a ket, we insert a label in the middle of this symbol, for instance, $|\psi\rangle$. Similarly, maps of the type $\mathcal{H} \to \mathbb{C}, |\varphi\rangle \mapsto \langle\cdot|\varphi\rangle$, where we obviously understand $\langle\cdot|\varphi\rangle = \langle|\cdot\rangle||\varphi\rangle\rangle$, are called "*bras*" and are denoted by the symbol $\langle\cdot|$. Again, if we want to specify a bra, we insert a label in the middle of this symbol, for instance $\langle\psi|$. Notice that $\langle\psi|$ is a vector of the dual space $\mathcal{H}^*$.

Definition 1.4.3
Let $\mathcal{H}$ be a Hilbert space.

(I) A *ray* is a class of the equivalence relation in $\mathcal{H}$

$$|\psi\rangle \sim |\varphi\rangle :\Leftrightarrow \exists\alpha \in \mathbb{C}_{\neq 0},\ |\psi\rangle = \alpha\,|\varphi\rangle$$

For any nonzero ray, we can choose a normalized representative of the class to which we will identify the ray in practice.

(II) An *operator* or *tensor* in $\mathcal{H}$ is a linear endomorphism $A\colon \mathcal{H} \to \mathcal{H}$. We denote by $Lin(\mathcal{H})$ the set of operators in $\mathcal{H}$. We can define a $\mathbb{C}-$vector space structure in $Lin(\mathcal{H})$ with the following operations:

(1) $(A+B)\,|\psi\rangle := A\,|\psi\rangle + B\,|\psi\rangle$

(2) $(\lambda A)\,|\psi\rangle := A\lambda\,|\psi\rangle \quad \forall\lambda \in \mathbb{C}$

Definition 1.4.4

Let $\mathcal{H}$ be a Hilbert space.

(I) Let $|\varphi_1\rangle, |\varphi_2\rangle \in \mathcal{H}$. The operator defined by

$$(|\varphi_1\rangle \otimes |\varphi_2\rangle)|\psi\rangle := \langle\psi|\varphi_2\rangle |\varphi_1\rangle \quad \forall |\psi\rangle \in \mathcal{H}$$

is called the *tensor product* of $|\psi_1\rangle$ and $|\psi_2\rangle$.

(II) A basis $\{|e_j\rangle\}$ of $\mathcal{H}$ is *orthonormal* if $\langle e_j|e_k\rangle = \delta_{jk} \quad \forall j,k$.

(III) Let $\mathcal{H}_1, \mathcal{H}_2$ be two Hilbert spaces with associated orthonormal basis, respect., $\{|j\rangle_1\}, \{|k\rangle_2\}$. We define the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$ as the Hilbert space with basis $\{|j\rangle_1 \otimes |k\rangle_2\}$ and inner product defined by

$$\left\langle |j\rangle_1 \otimes |k\rangle_2 \middle| \left|\hat{j}\right\rangle_1 \otimes \left|\hat{k}\right\rangle_2 \right\rangle_{12} := \delta_{j\hat{j}}\delta_{k\hat{k}}$$

Remark 1.4.5

$\otimes\colon \mathcal{H} \times \mathcal{H} \to Lin(\mathcal{H})$ is a bilinear map.

Notation 1.4.6(1) We will use $|j\rangle$ to denote the unit vector $|e_j\rangle$ of the canonical basis $(|e_j\rangle)_{j=0}^{N-1}$ of $\mathcal{H}$.

(2) Sometimes we may use $|\varphi_1, \varphi_2\rangle$ instead of $|\varphi_1\rangle \otimes |\varphi_2\rangle$ and $\mathcal{H}^{\otimes n}$ instead of $\mathcal{H} \otimes \overset{n}{\ldots} \otimes \mathcal{H}$.

Definition 1.4.7

Let $\mathcal{H}$ be a Hilbert space and $A\colon \mathcal{H} \to \mathcal{H}$ an operator.

(I) The adjoint A^* of the operator A is the operator defined by

$$\langle\varphi|A\psi\rangle = \langle A^*\varphi|\psi\rangle \quad \forall |\varphi\rangle, |\psi\rangle \in \mathcal{H}$$

where $|A\psi\rangle = A|\psi\rangle$ and $|A^*\psi\rangle = A^*|\psi\rangle$.

(II) A is a self-adjoint operator if $A = A^*$.

(III) A is an unitary operator if $AA^* = A^*A = I$, where I is the identity operator.

Remark 1.4.8

The existence and uniqueness of the adjoint operator is true in the finite dimensional Hilbert spaces. Also in this case, the eigenvectors of a self-adjoint operator A form a

orthonormal basis $\{|\psi_j\rangle\}_{j=1}^n$ as a result of the spectral theorem. We can express A as

$$A = \sum_{j=1}^{n} \lambda_j E_j$$

where λ_j is the j^{th} eigenvalue and E_j the j^{th} orthogonal projection. Since $E_j = |\psi_j\rangle \langle\psi_j|$, we can also write

$$A = \sum_{j=1}^{n} \lambda_j |\psi_j\rangle \langle\psi_j|$$

In the infinite case the definition of self-adjoint operator and the spectral theorem is more subtle.

Remark 1.4.9

We will only work with finite dimensional Hilbert spaces.

Remark 1.4.10

Let $\mathcal{H}$ be a finite dimensional Hilbert space, and let $\{|\psi_j\rangle\}_{j=1}^n$ be an orthonormal basis of $\mathcal{H}$. Let A be an operator with associated matrix $[A] = (a_{jk})$ in this basis. If $[A^*] = (b_{jk})$ is the associated matrix of A^*, by orthonormality we have

$$a_{jk} = \langle Ak|\psi_j\rangle$$
$$b_{jk} = \langle A^*k|\psi_j\rangle$$

Observe that

$$b_{jk} = \langle A^*\psi_k|\psi_j\rangle = \overline{\langle\psi_j|A^*\psi_k\rangle} = \overline{\langle A\psi_j|\psi_k\rangle} = \overline{a_{kj}}$$

where $\overline{\cdot}$ denotes the complex conjugation. Therefore $[A^*] = [A]^*$ follows, where the second $*$ denotes the adjoint matrix.

1.5 Postulates of Quantum Systems

The framework of quantum systems works in a similar way to the classical systems' one that we just saw, but with some key differences.

Postulate 1.5.1 (*States*)

In a quantum system, a state is a ray in a Hilbert space.

Remark 1.5.2

Since every ray corresponds to a possible state, given two states $|\psi\rangle$, $|\varphi\rangle$ another state can be constructed as the linear *superposition* of the two:

$$\alpha |\psi\rangle + \beta |\varphi\rangle$$

Postulate 1.5.3 (*Observables*)
In a quantum system, an observable is a self-adjoint operator.

Postulate 1.5.4 (*Measurement*)
In a quantum system, the measurement of an observable A prepares an eigenvector $|\psi_j\rangle$ of A, and the observer learns the value of the correspondent eigenspace ψ_j after the measurement. If the quantum state just prior to the measurement is $|\psi\rangle$, then the outcome ψ_j is obtained with *a priori* probability

$$\mathbb{P}_{\psi}(\psi_j) := \|E_j |\psi\rangle\|^2 = \||\psi_j\rangle \langle\psi_j|\psi\rangle\|^2 = |\langle\psi_j|\psi\rangle|^2$$

Postulate 1.5.5 (*Dynamics*)
The time evolution of a closed quantum system is described by a unitary operator.

Postulate 1.5.6 (*Composite Systems*)

(I) If the Hilbert spaces of two quantum system $\mathfrak{S}_1, \mathfrak{S}_2$ are, respect., $\mathcal{H}_1, \mathcal{H}_2$, then their composite quantum system $\mathfrak{S}_1\mathfrak{S}_2$ has for Hilbert space the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$.

(II) If the systems $\mathfrak{S}_1, \mathfrak{S}_2$ are prepared, respect., in the states $|\psi\rangle_1, |\psi\rangle_2$, then the composite system's state is the product $|\psi\rangle_1 \otimes |\psi\rangle_2$.

Definition 1.5.7
A closed system described by all the above postulates is a *quantum system*.

Chapter 2

Elements of Computing

All computers are based in the same principles: they use some input and a sequence of instructions to produce an output. We have followed the chapter 7 of [7] and chapters 3, 4 of [6].

2.1 Boolean Circuits

In this section we offer a model of classical computers: the *circuit model*[1]. A circuit is a physical device made up of *wires* and *logic gates*, which, respectively, carry information around and perform simple computational tasks. However, we can totally forget the physical substratum of logic gates and think of them abstractly as "elementary *Boolean functions*". This model is the more realistic one and the flavor of the ingredient for quantum computers.

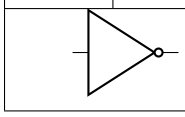
Definition 2.1.1 (I) A *Boolean function* is a function $f\colon \{0,1\}^m \to \{0,1\}^n$ from some number m of input bits to some number n of output bits.

(II) Let f be a Boolean function and $\mathbb{B}$ be a fixed set of Boolean functions. We call *Boolean circuit* of f in terms of the basis $\mathbb{B}$ a representation of f in terms of functions from $\mathbb{B}$.

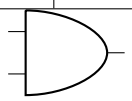
Example 2.1.2
Elementary Boolean functions and its corresponding circuit notation as a logic gate.

[1]The circuit model is equivalent to the Turing Machine model in terms of computational power, but it is less convenient for our approach in these notes.

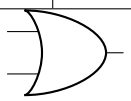
NOT($\neg$)	
Input	Output
0	1
1	0



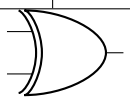
AND($\wedge$)		
Input		Output
0	0	0
0	1	0
1	0	0
1	1	1



OR($\vee$)		
Input		Output
0	0	0
0	1	1
1	0	1
1	1	1



XOR($\oplus$)		
Input		Output
0	0	0
0	1	1
1	0	1
1	1	0



Proposition 2.1.3

Any boolean function can be obtained as a combination of elements of the set of boolean functions $\{NOT, AND\}$.

Example 2.1.4

$\{NOT(\neg), AND(\wedge), OR(\vee)\}$ is redundant. Indeed with only NOT and AND we can obtain OR.

$$\neg(\neg A \wedge \neg B) = \neg(\neg A) \vee \neg(\neg B) = A \vee B$$

Remark 2.1.5

Logic gates are physical electronic circuits having the number of input and output electrodes as that of its corresponding elementary Boolean function. Bit values 0 and 1 correspond to physical voltages staying below or exceeding a certain voltage. Arbitrary Boolean functions are computed by physically connecting inputs and outputs of logic gates so that the resulting circuit implements the sought Boolean function. All the operations in a classical computer are in fact a very long sequence of logic gates.

2.2 Quantum Circuits

First, we define the quantum analog of the set of bits $\{0, 1\}$.

Definition 2.2.1

A *qubit* is a state of the quantum system defined by the Hilbert space $\mathcal{H} = \mathbb{C}^2$ with inner product $\langle\psi|\varphi\rangle = \psi_{(1)}\overline{\varphi}_{(1)} + \psi_{(2)}\overline{\varphi}_{(2)}$. A *$n-$qubit register* of $|\psi_1\rangle, \ldots, |\psi_n\rangle \in \mathbb{C}^2$ is the state $|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$ of the composite quantum system $\mathcal{H}^{\otimes n} = (\mathbb{C}^2)^{\otimes n}$.

Notation 2.2.2

(1) We denote $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ in $\mathbb{C}^2$ as in **1.4.6** (2).

(2) Notice that $\mathcal{H}^{\otimes t} = (\mathbb{C}^2)^{\otimes t}$ is a 2^t-dimensional vector space with canonical basis $\{|j_1\rangle \otimes \cdots \otimes |j_t\rangle\}_{j_1,\ldots,j_t \in \{0,1\}} = \{|j_1, \ldots, j_t\rangle\}_{j_1,\ldots,j_t \in \{0,1\}}$, so we denote this orthonormal basis naturally as $\{|j\rangle\}_{j=0}^{2^t-1}$.

(3) We will only work with the observable $\mathbf{A} := \sum_{j=0}^{2^t-1} |j\rangle \langle j|$ of $(\mathbb{C}^2)^{\otimes t}$.

Remark 2.2.3

An example of a physical realization of a qubit is an electron that orbits around an atom. An electron has a fundamental state and an excited state which we can represent, respect., with $|0\rangle$ and $|1\rangle$. There is an engineering method called *polarization* that makes possible changing the electron state $|0\rangle$ to $|1\rangle$ and vice versa. In practice, a qubit is a system very difficult to isolate and this is one of the problems of quantum information.

A classical or Boolean circuit implements a Boolean map f using logic gates. A quantum circuit implements a unitary operator $U\colon \mathbb{C}^2 \to \mathbb{C}^2$ using quantum gates, which are also unitary operators. This is a model for quantum computation in which a computation is a sequence of quantum gates.

Example 2.2.4

Elementary quantum gates.

Hadamard (H)
$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
$H\|j\rangle = \frac{1}{\sqrt{2}}((-1)^j \|j\rangle + \|1-j\rangle)\, \forall j \in \{0,1\}$

$\phi-$**phase** $(\Phi(\phi))$
$\begin{pmatrix} 1 & 0 \\ 0 & exp(2i\phi) \end{pmatrix}$
$\Phi(\phi)\|j\rangle = exp(2ij\phi)\, \forall j \in \{0,1\}$

NOT (N)
$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
$N\|j\rangle = \|1-j\rangle\ \forall j \in \{0,1\}$

Remark 2.2.5

It is straightforward that these three matrix are unitary.

"If A is true, then do B". This type of controlled operation is one of the most useful in computing.

Definition 2.2.6

Given an unitary operator $U\colon (\mathbb{C}^2)^{\otimes n} \to (\mathbb{C}^2)^{\otimes n}$, its *controlled–U operator* is the unitary operator $C(U)\colon \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes n} \to \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes n}$ defined by

$$C(U)\,|j\rangle \otimes |\psi\rangle := \begin{cases} |j\rangle \otimes |\psi\rangle & j = 0 \\ |j\rangle \otimes U\,|\psi\rangle & j = 1 \end{cases} \quad \forall\, |\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$$

Proposition 2.2.7

Any unitary operator can be approximated to arbitrary accuracy using the set or operators $\{H, \Phi(\frac{\pi}{4}), C(N), \Phi(\frac{\pi}{8})\}$.

Chapter 3

Shor's Factoring Algorithm

Shor's algorithm is a quantum algorithm that takes as input a composite integer N and returns a prime factor. We will deal with the case where $N = pq \in \mathbb{N}_{\geq 2}$ with p and q two prime numbers. The general case can easily be adapted from this one. We have followed the chapter 9 of [7] and chapter 5 of [6].

3.1 Introduction

The algorithm is inspired on the following two lemmas.

Lemma 3.1.1

Let $N = pq$ be a natural number product of two primes. Suppose that $x \in [\![1, N-1]\!]$ is a solution $\neq \pm 1 \pmod N$ to the equation $x^2 \equiv 1 \pmod N$. Then $gcd(x-1, N)$ and $gcd(x+1, N)$ are the non-trivial factors of N.

Proof. We have

$$x^2 \equiv 1 \pmod N \Rightarrow pq = N | (x-1)(x+1)$$

Now, $x \in [\![1, N-1]\!]$, so we have that $gcd(x-1, N)$ and $gcd(x+1, N)$ are factors of N, and they are $\neq 1$ because $x^2 \not\equiv \pm 1 \pmod N$. ■

Lemma 3.1.2

Let $N = pq$ be a natural number product of two primes and $x \in [\![1, N-1]\!]$ a uniformly chosen random integer such that $gcd(x, N) = 1$. Define the sets:

$$A := \{x \in [\![1, N-1]\!] : 2 | ord(x, N)\}$$
$$B := \{x \in [\![1, N-1]\!] : x^{\frac{ord(x,N)}{2}} \not\equiv -1 \pmod N\}$$

Then,

$$\mathbb{P}(A \cap B) \geq 1 - \frac{1}{4}$$

Proof. See [[6], *A*4.3]. ■

Definition 3.1.3
Let $x, N \in \mathbb{Z}_{\geq 2}$ such that $gcd(x, N) = 1$. We define the *order* of x in $\mathbb{Z}/N\mathbb{Z}$ by

$$ord(x, N) := inf\{r \geq 0 \colon x^r \equiv 1 \pmod{N}\}$$

The *order finding problem* consists in determine $ord(x, N)$.

A proposal of a classical factoring algorithm would be the following if we knew an efficient way to evaluate the function *ord*. However, with a classical computer there is no efficient way to do so.

Algorithm 3.1.4

(1) Pick a random number $x \in [\![1, N-1]\!]$ and compute $gcd(x, N)$. If $gcd(x, N) \neq 1$, we have the factorization. Else $gcd(x, N) = 1$ and we jump to step (2).

(2) Find $r = ord(x, N)$.

(3) If r is odd or $x^{\frac{r}{2}} \equiv -1 \pmod{N}$ go to the first step, else $gcd(x^{\frac{r}{2}} - 1)$ and $gcd(N, x^{\frac{r}{2}} + 1)$ are the non-trivial factors of N.

We will see in *section 3.4* a quantum algorithm that solves the order finding problem efficiently in a quantum computer. It requires two quantum circuits: *Quantum Fourier Transform* (QFT) and *Quantum Phase Estimation* (QPE).

3.2 Quantum Fourier Transform

Definition 3.2.1 (I) The *Discrete Fourier Transform* is the complex map

$$F \colon \mathbb{C}^n \to \mathbb{C}^n, x = (x_j)_{j=0}^{n-1} \mapsto F(x) = \Big(\frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} exp\Big(\frac{2\pi i k}{n} j\Big)\Big)_{j=0}^{n-1}$$

(II) The *Quantum Fourier Transform* is the operator defined by

$$\mathcal{F} \colon \mathbb{C}^n \to \mathbb{C}^n, |j\rangle \mapsto \mathcal{F}\,|j\rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} exp\Big(\frac{2\pi i k}{n} j\Big)\,|k\rangle \quad \forall j \in \{0, \dots, n-1\}$$

Lemma 3.2.2

The operator $\mathcal{F}$ is unitary.

Proof. We describe $\mathcal{F}$ and $\mathcal{F}^*$ in terms of the dual basis:

$$\mathcal{F} = \frac{1}{\sqrt{n}} \sum_{j,k=0}^{n-1} exp(\frac{2\pi ik}{n} j) \left|k\right\rangle \left\langle j\right|$$

$$\mathcal{F}^* = \frac{1}{\sqrt{n}} \sum_{j',k'=0}^{n-1} exp(\frac{-2\pi ik'}{n} j') \left|k'\right\rangle \left\langle j'\right|$$

Then,

$$\mathcal{F}^*\mathcal{F} = \frac{1}{n} \sum_{j',k',j,k=0}^{n-1} exp(\frac{2\pi i(j'k' - jk)}{n} j) \left|j\right\rangle \left\langle k|k'\right\rangle \left\langle j'\right|$$

Now, using the identities

$$\left\langle k|k'\right\rangle = \delta_{kk'}$$

$$\frac{1}{n} \sum_{k=0}^{n-1} exp(\frac{2\pi ik(j-j')}{n}) = \delta_{jj'}$$

we get

$$\mathcal{F}\mathcal{F}^* = 1 = \mathcal{F}^*\mathcal{F}$$

■

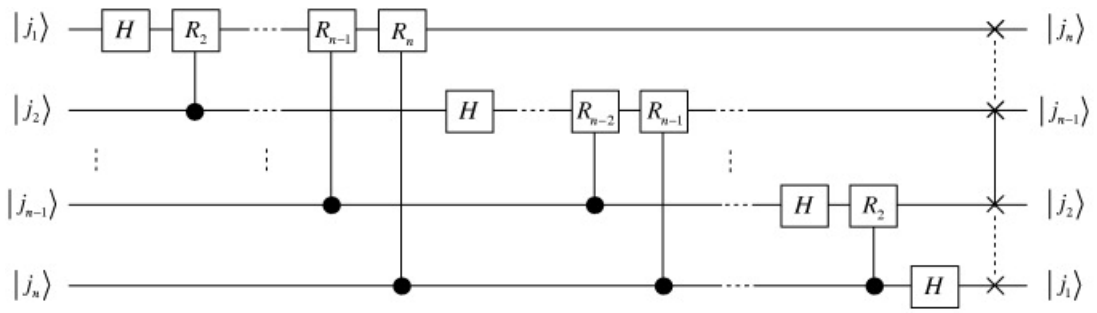


Figure 3.1: **QFT circuit**. *The input is on the left and the output on the right. The H blocks represent Hadamard gates and the R_k blocks represent $C(\Phi(\frac{\pi}{2^k}))$ gates controlled by the qubit with the corresponding bullet.*

Lemma 3.2.3

We can build a quantum circuit for the Quantum Fourier Transform from Hadamard gates and controlled phase gates, as in figure **3.1**, and similarly for its adjoint operator if

we invert this circuit (i.e., reversing the order of the gates and taking the adjoint of each gate).

3.3 Quantum Phase Estimation

Let $n \in \mathbb{N}_{\geq 1}$ be arbitrary fixed. Let $U \colon (\mathbb{C}^2)^{\otimes n} \to (\mathbb{C}^2)^{\otimes n}$ be a unitary operator and $|u\rangle \in (\mathbb{C}^2)^{\otimes n}$ an eigenvector of U. The *quantum phase estimation problem* consists in determine $\phi_u \in [0, 1]$ such that

$$U\,|u\rangle = exp(2\pi i \phi_u)\,|u\rangle$$

Remark 3.3.1
Recall from **2.2.2** that $(\mathbb{C}^2)^{\otimes n}$ is a Hilbert space of dimension 2^n.

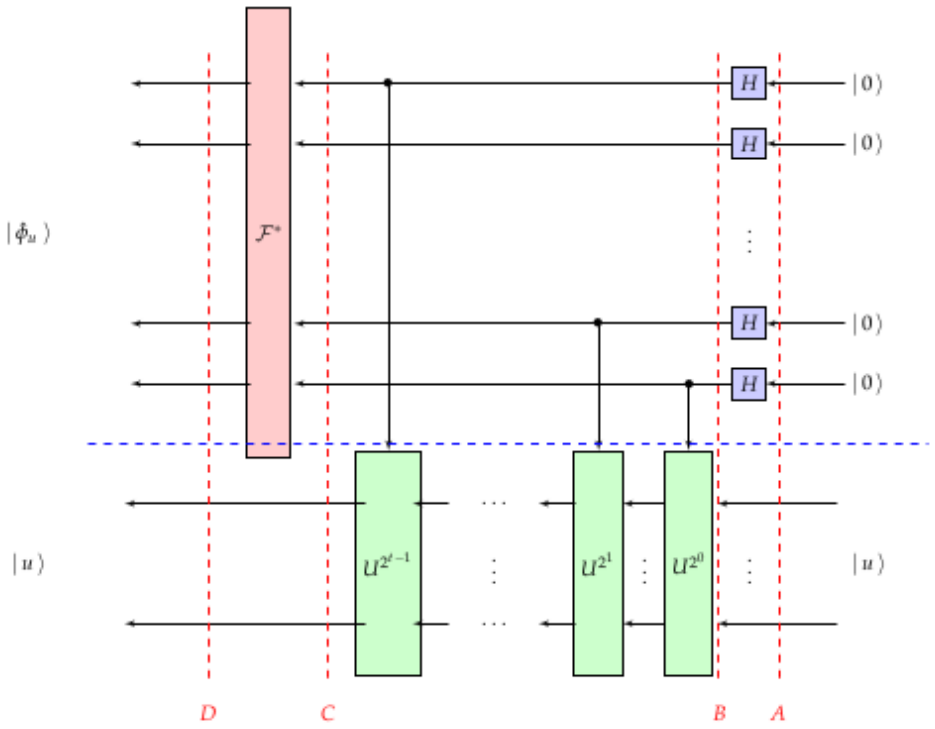


Figure 3.2: **QPE circuit**. *The input is on the right and the output on the left. The pink $\mathcal{F}^*$ block represent the QFT adjoint circuit, the blue H blocks represent Hadamard gates and the green U^{2^k} blocks represent $C(U^{2^k})$) gates controlled by the qubit with the corresponding bullet.*

Consider the quantum circuit of figure **3.2**. The input of the circuit is composed by a 1^{st} $t-$qubit register $|0\rangle^{\otimes t} \in (\mathbb{C}^2)^{\otimes t}$ tensored with a 2^{nd} $n-$qubit register $|u\rangle \in (\mathbb{C}^2)^{\otimes n}$

(the eigenvector of U). We will describe how can we obtain ϕ_u accurate to some $\nu \in \mathbb{N}$ bits with large probability.

Proposition 3.3.2

$$\forall \varepsilon \in]0,1[\ \ \exists t(\varepsilon) \in \mathbb{N}_{\geq 1} \colon \mathbb{P}_{\hat{\phi}_u}(\Delta) \geq 1 - \varepsilon$$

where $\Delta = \{l \in [\![0, 2^t - 1]\!] \colon |\frac{l}{2^t} - \phi_u| \leq 2^{-\nu}\}$ and $\left|\hat{\phi}_u\right\rangle$ is the state of the 1^{st} register at the stage D.

Proof. We will compute the state of the register through the stages A, B, C, D using the bilinearity of $\otimes$ and after that we will make a measurement.

(1) Stage A: *Initialisation.*

We initialize the register to:

$$\begin{aligned} |\psi_{A1}\rangle &= |0\rangle^{\otimes t} \\ |\psi_{A2}\rangle &= |u\rangle \\ |\psi_A\rangle &= |\psi_{A1}\rangle \otimes |\psi_{A2}\rangle = |0\rangle^{\otimes t} \otimes |u\rangle \end{aligned}$$

(2) Stage B: *Hadamard Gates.*

Remember that, according to the definition, $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Then, at B the states are:

$$\begin{aligned} |\psi_{B1}\rangle &= (H|0\rangle)^{\otimes t} = (\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))^{\otimes t} \\ |\psi_{B2}\rangle &= |u\rangle \\ |\psi_B\rangle &= |\psi_{B1}\rangle \otimes |\psi_{B2}\rangle = (\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))^{\otimes t} \otimes |u\rangle \end{aligned}$$

(3) Stage C: *Controlled Gates.*

Denote by $\left|\psi_{B1}^{(k)}\right\rangle$ the k^{th} qubit of the state $|\psi_{B1}\rangle$

Observe that, for all $j \in \{0, ..., t-1\}$:

$$\begin{aligned} &C(U^{2^j})\left|\psi_{B1}^{(t-j)}\right\rangle \otimes |\psi_{B2}\rangle = \\ &= C(U^{2^j})(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) \otimes |u\rangle = C(U^{2^j})\frac{1}{\sqrt{2}}(|0\rangle \otimes |u\rangle + |1\rangle \otimes |u\rangle) \\ &= \frac{1}{\sqrt{2}}(C(U^{2^j})|0\rangle \otimes |u\rangle + C(U^{2^j})|1\rangle \otimes |u\rangle) = \frac{1}{\sqrt{2}}(|0\rangle \otimes |u\rangle + |1\rangle \otimes U^{2^j}|u\rangle) \\ &= \frac{1}{\sqrt{2}}(|0\rangle \otimes |u\rangle + |1\rangle \otimes exp(2\pi i 2^j \phi_u)|u\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + exp(2\pi i 2^j \phi_u)|1\rangle) \otimes |u\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle + exp(2\pi i 2^j \phi_u)|1\rangle) \otimes |\psi_{B2}\rangle \end{aligned}$$

and therefore the second register remains unaltered whenever we apply the gate $C(U^{2^j})$ in this way thanks to the fact that u is an eigenvector of U.

Now, suppose we want to apply sequentially the gates $C(U^{2^0}), C(U^{2^1}), \ldots, C(U^{2^{t-1}})$ to the state $|\psi_B\rangle$ in order to obtain a new state $|\psi_C\rangle$, as in the figure. We get

$$\left|\psi_{C1}^{(t-j)}\right\rangle = \frac{1}{\sqrt{2}}(|0\rangle + exp(2\pi i 2^j \phi_u)\,|1\rangle) \quad \forall j \in \{0, \ldots, t-1\}$$

and therefore

$$|\psi_{C1}\rangle = \frac{1}{2^{\frac{t}{2}}} \bigotimes_{j=0}^{t-1} (|0\rangle + exp(2\pi i 2^j \phi_u)\,|1\rangle)$$

since $|\psi_{B2}\rangle = |u\rangle$. Of course,

$$|\psi_{C2}\rangle = |\psi_{B2}\rangle = |u\rangle$$

so

$$\begin{aligned}
|\psi_C\rangle = |\psi_{C1}\rangle \otimes |\psi_{C2}\rangle &= \frac{1}{2^{\frac{t}{2}}} \bigotimes_{j=0}^{t-1} (|0\rangle + exp(2\pi i 2^j \phi_u)\,|1\rangle) \otimes |u\rangle \\
&= \frac{1}{2^{\frac{t}{2}}} \sum_{k_1,\ldots,k_{t-1} \in \{0,1\}} exp(2\pi i \phi_u \langle k_{t-1} \ldots k_0\rangle_2)\,|k_{t-1} \ldots k_0\rangle \otimes |u\rangle \\
&= \frac{1}{2^{\frac{t}{2}}} \sum_{k=0}^{2^t-1} exp(2\pi i \phi_u k)\,|k\rangle \otimes |u\rangle
\end{aligned}$$

where in the second last equality we have used the bilinearity of $\otimes$.

(4) Stage D: *Adjoint Quantum Fourier Transform.*

Now, we apply $\mathcal{F}^*$ to the 1^{st} register. The 2^{nd} register remains the same.

$$\begin{aligned}
|\psi_{D1}\rangle = \mathcal{F}^*\,|\psi_{C1}\rangle &= \frac{1}{2^{\frac{t}{2}}} \sum_{k=0}^{2^t-1} exp(2\pi i \phi_u k) \mathcal{F}^*\,|k\rangle \\
&= \frac{1}{2^{\frac{t}{2}}} \sum_{k=0}^{2^t-1} [exp(2\pi i \phi_u k) \frac{1}{2^{\frac{t}{2}}} \sum_{l=0}^{2^t-1} exp\Big(\frac{-2\pi i l}{2^t} k\Big)\,|l\rangle] \\
&= \frac{1}{2^t} \sum_{l,k=0}^{2^t-1} exp(2\pi i \phi_u k) exp\Big(\frac{-2\pi i l}{2^t} k\Big)\,|l\rangle \\
|\psi_{D2}\rangle &= |u\rangle \\
|\psi_D\rangle = |\psi_{D1}\rangle \otimes |\psi_{D2}\rangle &= \frac{1}{2^t} \sum_{l,k=0}^{2^t-1} exp(2\pi i \phi_u k) exp\Big(\frac{-2\pi i l}{2^t} k\Big)\,|l\rangle \otimes |u\rangle
\end{aligned}$$

(5) Measurement. Consider the observable **A** of **2.2.2** (3). Now we make a measurement on the observable **A** at the current state $|\psi_{D1}\rangle$ of the 1^{st} register, but in a special way.

Let $L \in [\![0, 2^t - 1]\!]$ fixed and denote $\left|\hat{\phi_u}\right\rangle = |\psi_{D1}\rangle$. Observe that

$$\{l \in [\![0, 2^t - 1]\!]\} = \{(l + L) \mod 2^t : l \in [\![-2^{t-1} + 1, 2^{t-1}]\!]\}$$

so

$$\begin{aligned}\left|\hat{\phi_u}\right\rangle &= \frac{1}{2^t} \sum_{l=-2^{t-1}+1}^{2^{t-1}} \sum_{k=0}^{2^t-1} exp(2\pi i \phi_u k) exp(\frac{-2\pi i((l+L) \mod 2^t)}{2^t} k) \left|(l + L) \mod 2^t\right\rangle \\ &= \sum_{l=-2^{t-1}+1}^{2^{t-1}} [\sum_{k=0}^{2^t-1} \frac{1}{2^t} exp(2\pi i(\phi_u - \frac{l+L}{2^t}))^k] \left|(l + L) \mod 2^t\right\rangle\end{aligned}$$

Define $\forall l \in [\![-2^{t-1} + 1, 2^{t-1}]\!]$

$$\alpha_l := \sum_{k=0}^{2^t-1} \frac{1}{2^t} exp\Big(2\pi i\Big(\phi_u - \frac{l+L}{2^t}\Big)\Big)^k = \frac{1}{2^t}\Big(\frac{1 - exp(2\pi i(2^t\phi_u - (l+L)))}{1 - exp(2\pi i(\phi_u - \frac{l+L}{2^t}))}\Big) \quad (3.1)$$

because it is a geometric series and recall from **1.5.4** that

$$\mathbb{P}_{\hat{\phi_u}}((l + L) \mod 2^t) = |\alpha_l|^2 \quad \forall l \in [\![-2^{t-1} + 1, 2^{t-1}]\!]$$

(6) Estimation of ϕ_u. Let $L \in [\![0, 2^t - 1]\!]$ and $e \in \mathbb{N}_{\geq 1}$ fixed. Define:

$$\begin{aligned}\Delta(L, e) &:= \{l \in [\![0, 2^t - 1]\!] : |l - L| > e\} \\ &= \{(l + L) \mod 2^t : l \in [\![-2^{t-1} + 1, 2^{t-1}]\!], |[(l + L) \mod 2^t] - L| > e\} \\ &= \{(l + L) \mod 2^t : l \in [\![-2^{t-1} + 1, 2^{t-1}]\!], |l| > e\}\end{aligned}$$

After the measurement previous part,

$$\mathbb{P}_{\hat{\phi_u}}(\Delta(L, e)) = \sum_{l=-2^{t-1}+1}^{-(e+1)} |\alpha_l|^2 + \sum_{l=e+1}^{2^{t-1}} |\alpha_l|^2$$

• Suppose that L is such that $\frac{L}{2^t} = \langle 0.L_1 \dots L_t\rangle_2$ is the best $t-$bit approximation to ϕ_u which is less than ϕ_u, i.e., $\delta := \phi_u - \frac{L}{2^t}$ is in $[0, 2^{-t}]$.

Observe that we have $\frac{2|\theta|}{\pi} \leq \|1 - exp(i\theta)\| \leq 2 \; \forall \theta \in [0, \pi]$ where the two inequalities holds due to elementary calculus : $\|1 - exp(i\theta)\|^2 = 2(1 - cos(\theta))$, then the second inequality is trivial and for the first one we consider the function $f(\theta) = 4\frac{\theta^2}{\pi} + 2cos(\theta) - 2$ and analyze which signs takes the derivative in $[-\pi, \pi]$.

In particular, $l \in [-2^{t-1}, 2^{t-1}]$ implies $2\pi(\delta - \frac{l}{2^t}) \in [-\pi, \pi]$. Thus after equation **3.1**

$$|\alpha_l| \leq \frac{1}{2^{t+1}(\delta - \frac{l}{2^t})}$$

and

$$\mathbb{P}_{\hat{\phi}_u}(\Delta(L,e)) \leq \frac{1}{4}[\sum_{l=-2^{t-1}+1}^{-(e+1)} \frac{1}{(l-2^t\delta)^2} + \sum_{l=e+1}^{2^{t-1}} \frac{1}{(l-2^t\delta)^2}]$$

Since by hypothesis $\delta \in [0, 2^{t-1}]$, we have:

– If $l \in [\![-2^{t-1}+1, -(e+1)]\!]$,

$$-2^{t-1} \leq l-1 \leq l-2^t\delta \leq l \leq -1 \Rightarrow (l-2^t\delta)^2 \geq l^2$$

– If $l \in [\![e+1, 2^{t-1}]\!]$,

$$0 \leq l-1 \leq l-2^t\delta \leq l \leq 2^{t-1} \Rightarrow (l-2^t\delta)^2 \geq (l-1)^2$$

and therefore

$$\mathbb{P}_{\hat{\phi}_u}(\Delta(L,e)) \leq \frac{1}{4}[\sum_{l=-2^{t-1}+1}^{-(e+1)} \frac{1}{l^2} + \sum_{l=e+1}^{2^{t-1}} \frac{1}{(l-1)^2}] \leq \frac{1}{2}\sum_{l=e}^{2^{t-1}-1} \frac{1}{l^2} \leq \int_{e-1}^{2^{t-1}-1} \frac{1}{l^2} dl \leq \frac{1}{2(e-1)}$$

• Suppose we wish to approximate ϕ_u with accuracy $2^{-\nu}$, i.e., we choose $e = 2^{t-\nu}-1$. Then the probability of obtaining an approximation correct to this accuracy is at least

$$\mathbb{P}_{\hat{\phi}_u}([\![0, 2^t-1]\!] - \Delta(L,e)) \geq 1 - \frac{1}{2(2^{t-\nu}-1-1)}$$

Finally, if we want to successfully obtain ϕ_u accurate to ν bits with probability $1-\varepsilon \;\; \forall \varepsilon \in]0,1[$ we have to choose

$$t := \nu + \lceil log_2\Big(2 + \frac{1}{2\varepsilon}\Big) \rceil$$

■

3.4 Order Finding Algorithm

We will build an algorithm based in the *QPE* circuit choosing a special operator U and a special 1^{st} register $|u\rangle$.

Definition 3.4.1
Let $x, N \in \mathbb{Z}_{\geq 2}$ such that $pgcd(x, N) = 1$ and let $n = \lceil logN \rceil$. We define the *order finding operator* of x modulo N to be the mapping

$$OF_{x,N} \colon (\mathbb{C}^2)^{\otimes n} \to (C^2)^{\otimes n}, |y\rangle \mapsto OF_{x,N} |y\rangle := \begin{cases} |xy \mod N\rangle & y \in [\![0, N-1]\!] \\ |y\rangle & y \in [\![N, 2^n - 1]\!] \end{cases}$$

Remark 3.4.2
$OF_{x,N}$ is an unitary operator ($\langle x, y_1 | x, y_2\rangle = \langle x|x\rangle \langle y_1|y_2\rangle = \langle y_1|y_2\rangle$ since $|x\rangle$ can be chosen to be normalized).

Proposition 3.4.3
Let $r = ord(x, N)$, $n = \lceil logN \rceil$ and define

$$|u_s\rangle := \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} exp(-\frac{2\pi iks}{r}) \left|x^k \mod N\right\rangle \in (\mathbb{C}^2)^{\otimes n} \quad \forall s \in [\![0, r-1]\!]$$

Then

(i) $OF_{x,N} |u_s\rangle = exp(\frac{2\pi is}{r}) |u_s\rangle$

(ii) $\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle$

Proof. (i)

$$\begin{aligned} OF_{x,N} |u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} exp(\frac{-2\pi iks}{r}) \left|x^{k+1} \mod N\right\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{k=1}^{r} exp(\frac{-2\pi i(k-1)s}{r}) \left|x^k \mod N\right\rangle \\ &= exp(\frac{2\pi is}{r}) \frac{1}{\sqrt{r}} (\sum_{k=1}^{r-1} exp(\frac{-2\pi iks}{r}) \left|x^k \mod N\right\rangle + |x^r \mod N\rangle) \\ &= exp(\frac{2\pi is}{r}) |u_s\rangle \end{aligned}$$

(ii) Since $exp(\frac{-2\pi iks}{r})$ is a r^{th} root of the unity, $\sum_{s=0}^{r-1} exp(\frac{-2\pi iks}{r}) = r\delta_{k0}$ and

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = \frac{1}{r}\sum_{k,s=0}^{r-1} exp(\frac{-2\pi iks}{r})\,|x^k \mod N\rangle$$
$$= \frac{1}{r}\sum_{k=0}^{r-1} r\delta_{k0}\,|x^k \mod N\rangle = |1\rangle$$

■

Remark 3.4.4

Proposition **3.4.3** (i) establishes that $|u_s\rangle$ supposes previous knowledge of r. **3.4.3** (ii) fixes this because it suggests that we can choose $|u\rangle = |1\rangle$ in the QPE.

If we are able to deduce $\frac{s}{r}$ for different values of s, we shall be able to estimate r and the problem of order finding will be solved. This estimation is solved with the classical algorithm of continued fractions.

3.4.1 Continued Fraction Expansion

The following results about continued fractions can be checked in [3].

Definition 3.4.5

A *continued fraction expansion* (CFE) is an expression of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots}}}$$

where $a_0 \in \mathbb{N}$ and $a_m \in \mathbb{N}_{\geq 1}\ \forall m \geq 1$. We will write $[a_0; a_1, a_2, \dots]$ instead of the above fraction.

Let

$$p_m(\alpha) := \begin{cases} 1 & m = -1 \\ a_0 & m = 0 \\ a_m p_{m-1} + p_{m-2} & m \geq 1 \end{cases} \qquad q_m(\alpha) := \begin{cases} 0 & m = -1 \\ 1 & m = 0 \\ a_m q_{m-1} + q_{m-2} & m \geq 1 \end{cases}$$

The sequence $(\frac{p_m(\alpha)}{q_m(\alpha)})_{m\geq 0}$ is called the *sequence of principal convergents* of α.

Remark 3.4.6

(1) If $\alpha \in \mathbb{R}_{\geq 0}$, we can decompose it in a *CFE* $[a_0; a_1, a_2, \dots]$. We will write $\alpha =$

$[a_0; a_1, a_2, \dots]$. Furthermore, $\forall m \geq 0$ we have $[a_0; a_1, \dots, a_m] = \frac{p_m(\alpha)}{q_m(\alpha)}$, so this ratios are a rational approximation of α.

(2) Let $\alpha \in \mathbb{R}_{\geq 0}$. If $\alpha \in \mathbb{Q}_{>0}$, then there are exactly two *CFE* associated to α and they are both finite, namely $[a_0; a_1, \dots, a_M, 1]$ and $[a_0; a_1, \dots, a_{M-1}, a_M + 1]$ for some $M \geq 1$. Otherwise, if $\alpha \in \mathbb{R}_{\geq 0} - \mathbb{Q}_{>0}$, then α has a unique *CFE* and it is infinite.

Lemma 3.4.7

Let $\alpha = [a_0; a_1, a_2, \dots]$.

(i) $\lim_{m \to \infty} (\frac{p_m(\alpha)}{q_m(\alpha)}) = \alpha$

(ii) If $|\alpha - \frac{p}{q}| \leq \frac{1}{2q^2}$ for some $\frac{p}{q} \in \mathbb{Q}$ such that $q > 0$ and $pgcd(p, q) = 1$, then

$$\left|\alpha - \frac{p_m(\alpha)}{q_m(\alpha)}\right| \leq \frac{1}{2q_m(\alpha)^2} \quad \forall m \in \mathbb{N}_{\geq 1}$$

i.e, exists $M \in \mathbb{N}$ such that $\frac{p}{q} = \frac{p_M(\alpha)}{q_M(\alpha)}$.

Algorithm 3.4.8 (CFE)

Require: $\alpha \in \mathbb{R}_{>0}$, $M \in \mathbb{Z}_{>0}$
Ensure: $a_0, \dots, a_M$ such that $[a_0; a_1, \dots, a_m] = \frac{p_m(\alpha)}{q_m(\alpha)}$

```
m ← 0
while m ≤ M do
    a_m ← ⌊α⌋
    β ← α − ⌊α⌋
    m ← m + 1
    if β ≠ 0 then
        α ← 1/β
    else
        α ← 0
    end if
end while
```

3.4.2 Algorithm

Now let $x, N \in \mathbb{Z}_{\geq 2}$ such that $pgcd(x, N) = 1$ and $r = ord(x, N)$. Fix $\varepsilon \in]0, 1[$ and $s \in [\![0, r-1]\!]$. Consider the QPE circuit of figure **3.2** such that

$$\begin{aligned} n &:= \lceil logN \rceil \\ t &:= (2n+1) + \lceil log_2(2 + \frac{1}{2\varepsilon}) \rceil \\ U &:= OF_{x,N} \\ |u\rangle &:= |u_s\rangle \\ \phi_u &:= \frac{s}{r} \end{aligned}$$

It is clear from proposition **3.3.2** that making a measurement at the state $|\psi_{D1}\rangle = \left|\hat{\phi}_u\right\rangle$ provides with a $(2n+1)-$ bit approximation $\hat{\phi}_u$ of $\frac{s}{r}$ with probability at least $1 - \varepsilon$.

If we choose instead $|u\rangle := |1\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle$ and let $\phi_{u_s} = \frac{s}{r} \quad \forall s \in [\![0, r-1]\!]$, making a measurement at the state $|\psi_{D1}\rangle = \left|\hat{\phi}_u\right\rangle$ provides with a $(2n+1)-$ bit approximation $\hat{\phi}_{u_s}$ of a $\frac{s}{r}$ with probability at least $\frac{1-\varepsilon}{r}$. (Now $\left|\hat{\phi}_u\right\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \left|\hat{\phi}_{u_s}\right\rangle$ is a superposition with amplitudes $\frac{1}{r}$ and we know the case $|u\rangle := |u_s\rangle$ from proposition **3.3.2**). We denote this last QPE circuit by $QPE_{x,N}$.

Proposition 3.4.9

Applying the CFE algorithm on this approximation determines the order r with large probability.

Proof. Suppose $\hat{\phi}_{u_s}$ is the $(2n+1)-$bit approximation of a $\frac{s}{r}$. It follows that

$$|\hat{\phi}_{u_s} - \frac{s}{r}| \leq 2^{-(2n+1)} \leq \frac{1}{2r^2}$$

and by lemma **3.4.7** $\frac{s}{r} = CFE(\hat{\phi}_{u_s}) = \frac{p_M(\hat{\phi}_{u_s})}{q_M(\hat{\phi}_{u_s})}$ such that $pgcd(p_M(\alpha), q_M(\alpha)) = 1$, so if $x^{q_M(\hat{\phi}_{u_s})} \equiv 1 \pmod{N}$, we have $ord(x, N) = r = q_M(\hat{\phi}_{u_s})$. It is clear that this happens with large probability. ■

Algorithm 3.4.10 (OF)

Require: $x, N \in \mathbb{Z}_{\geq 2}$ such that $pgcd(x, N) = 1$
Ensure: $ord(x, N)$ with probability $1 - \varepsilon$
Set up the $QPE_{x,N}$ circuit
$\theta \leftarrow QPE_{x,N}$ (a $(2n+1)-$bit approximation of a $\frac{s}{r}$)
$\alpha := [a_0; a_1, \ldots, a_m] \leftarrow CFE(\theta)$
$\frac{s}{r} \leftarrow \frac{p_m(\alpha)}{q_m(\alpha)}$

if $x^r \equiv 1 \pmod{N}$ **then**
  return r
else
  return "The algorithm fails"
end if

Remark 3.4.11

The algorithm fails if:

(1) It produces a false estimate of a $\frac{s}{r}$. This happens with probability ε.

(2) s and r have a common factor. Then the element determined by the algorithm is a factor of r.

Fortunately, we have the following result

Proposition 3.4.12

The probability that r is the correct order is $\geq \frac{1}{4}$ and can be arbitrarily improved at the expense of several independent repetitions of the $QPE_{x,N}$ circuit.

Proof. See proposition 9.3.9 of [7]. ■

3.5 Shor's Algorithm

Finally, we resume Shor's Algorithm as follows:

Algorithm 3.5.1 (SHOR)

Require: $N \in \mathbb{Z}_{>0}$ of n bits such that $N = pq$ with p, q primes, $\varepsilon \in]0, 1[$.
Ensure: p, q with probability $1 - \varepsilon$.
  Set up the $QPE_{x,N}$ circuit with the given ε.
  $x, r \leftarrow 1$
  while r is odd **or** $x^{\frac{r}{2}} \equiv -1 \pmod{N}$ **do**
    $x \leftarrow (\llbracket 1, N-1 \rrbracket).\text{random\_element}()$
    $r \leftarrow OF(x, N)$
  end while
  if $r ==$"The algorithm fails" **then**
    return r
  else
    $p \leftarrow gcd(N, x^{\frac{r}{2}} - 1)$
    $q \leftarrow gcd(N, x^{\frac{r}{2}} + 1)$
    return p, q
  end if

Appendix A

C Language Implementations

We have made some implementations of the classical parts of the algorithm, including the CFE and the computation of the factors once we have the order. Also, we show some exemples of the manipulation of operators.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
//periode de a
//test r pair et a/2 +1
// return pgcd a/2 +1 et -1

/*Receive a and b
 * Return a^b
*/
unsigned int puissance(unsigned int a,unsigned int b,unsigned int n){
    unsigned int res = 1;
    int i;
    for(i=0;i<b;i++){
        res = (res*a)%n;
    }
    return res;
}
/*Receive a and b
 * return the gcd between a and b
*/
unsigned int pgcd(unsigned int a,unsigned int b){
    unsigned int c = 1;
    while (1){
        c = a%b;
        if (c == 0){
```

```
28              return b;
29          }
30          a = b;
31          b = c;
32      }
33  }
34  /*Receive a and b
35   *return a random unsigned int between a and b
36  */
37
38  unsigned int rand_a_b(unsigned int a, unsigned int b){
39      srand(time(NULL));
40      return rand()%(b-a) + a;
41  }
42
43  /*Receive a and n
44   * return the period of a mod n
45  */
46  unsigned int period(unsigned int a, unsigned int n){
47      unsigned int r = 1;
48      unsigned int res;
49      res = a%n;
50      while (res != 1){
51          r+=1;
52          res = (res*a)%n;
53      }
54      return r;
55  }
56
57  /*Receive n (n=p*q with p and q primes numbers)
58   * return p
59  */
60
61  unsigned int shor(unsigned int n){
62      unsigned int a,r,i;
63      i = 1;
64      //Step 1
65      do{
66          a = rand_a_b(2,n-1);
67          if (pgcd(a,n) != 1){
68              return pgcd(a,n);
69          }
70          //Step 2
71          r = period(a,n);
72          i = i +1;
```

```
73      }
74      while(((r%2) != 0 ) || ( (puissance(a,r/2,n) + 1) %n == 0));
            //Step 3
75      printf("a = %u \nperiode = %u",a,r);
76      //Step 4
77      return pgcd(puissance(a,r/2,n) + 1, n);
78  }
79
80  /*Receive a and b return the inverse of a mod b*/
81  unsigned int extended_euclide(unsigned int a, unsigned int b){
82      int r0=a,r1=b,r2, u2, v2,u0=1,u1=0, v0=0,v1=1,q1;
83
84      while (r1!=0){
85              q1=r0/r1;
86              r2=r0%r1;
87              u2=u0-q1*u1;
88              v2=v0-q1*v1;
89              r0=r1; r1=r2;
90              u0=u1; u1=u2;
91              v0=v1; v1=v2;
92      }
93      while (u0<0){
94          u0+=b;
95      }
96      return u0;
97  }
98
99
100
101 /* Main program*/
102 int main(void){
103     unsigned int p,q,n,e;
104     printf("Valeur de n : ");
105     scanf("%u",&n);
106     printf("Valeur de e : ");
107     scanf("%u",&e);
108     clock_t t;
109     p = shor(n);
110     t = clock();
111     q = n/p;
112     printf("p = %u \nq = %u\n",p,q);
113     printf("d = %u\n", extended_euclide(e,(p-1)*(q-1)));
114     printf("effectue en %f secondes ! \n",(float) t/CLOCKS_PER_SEC);
115
116
```

```
117     return 0;
118 }
```

shor.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4
5  //Qbits part
       ------------------------------------------------------------
6  typedef struct {
7      int n; //Size of the qbits
8      int *bits;
9      double *poids;
10 }qbits;
11
12 /*Create a qbits and allocate the memory needed*/
13 void create_qbits(qbits *q, int l){
14     q->n = l;
15     q->bits = (int*) calloc(q->n ,sizeof(int));
16     q->poids = (double*) calloc(q->n ,sizeof(double));
17 }
18
19 /*Receive a qbits and two lists (and l the len of this two lists),
       fill the qbits with the data in the lists*/
20 void fill_qbits(qbits *q, double *p, int *b, int l){
21     int i;
22     if(l != q->n){
23         printf("fill_qbits failed: Wrong size of list\n");
24         return;
25     }
26     for(i=0;i<q->n;i++){
27         q->poids[i] = p[i];
28         q->bits[i] = b[i];
29     }
30     return;
31 }
32
33 /*Receive a qbits and print it*/
34 void print_qbits(qbits q){
35     int i;
36     printf("%f |%d>",q.poids[0],q.bits[0]);
37     for(i=1; i<q.n;i++){
38         if(q.poids[i] > 0){
```

```
39              printf(" + %f |%d>",q.poids[i],q.bits[i]);
40          }
41          else{
42              printf(" - %f |%d>",abs(q.poids[i]),q.bits[i]);
43          }
44      }
45      printf("\n");
46      return;
47  }
48
49
50  //Matrix part
        ------------------------------------------------------------
51  typedef struct {
52      int nrows;
53      int ncol;
54      double *vector;
55  }Matrix;
56
57  /*Create a Matrix, and allocate the memory needed*/
58  void create_matrix(Matrix * mat, int nr, int nc){
59      mat->nrows = nr;
60      mat->ncol = nc;
61      mat->vector = (double*) calloc(mat->nrows *
            mat->ncol,sizeof(double));
62      return;
63  }
64
65  /*
66  void init_matrix(Matrix *mat){
67      int i,j;
68      for(i=0; i<mat->nrows; i++){
69          for(j=0; j<mat->ncol; j++){
70              mat->vector[i*mat->ncol + j] = 1+i+j;
71          }
72      }
73      return;
74  }
75  */
76
77  /*Receive a Matrix, print it*/
78  void print_matrix(Matrix mat){
79      int i;
80      printf("Matrice with %d rows and %d columns \n",mat.nrows,
            mat.ncol);
```

```
81      for(i=0; i<mat.nrows* mat.ncol;i++){
82          if((i%mat.ncol) ==0){
83              printf("\n");
84          }
85          printf("%f",mat.vector[i]);
86      }
87      printf("\n");
88      return;
89  }
90
91  /*Receive A, B and &C two Matrix and one adress of a Matrix
92   * Compute A*B and store it in &C*/
93  void mult_matrix(Matrix A, Matrix B, Matrix *C){
94      int i,j,k;
95      if(A.ncol != B.nrows){
96          printf("Not possible");
97          return;
98      }
99      C->ncol = B.ncol;
100     C->nrows = A.nrows;
101     C->vector = (double*) calloc(C->nrows * C->ncol, sizeof(double));
102     for(i = 0; i<C->nrows; i++){
103         for(j=0; j<C->ncol; j++){
104             for(k=0; k<A.ncol; k++){
105                 C->vector[i*C->ncol + j] += A.vector[i*A.ncol + k] *
                        B.vector[k*C->ncol + j];
106             }
107         }
108     }
109     return;
110 }
111
112 /*Create some basic logcical gate (Hadamard gate ...*/
113 void create_logical_gate(Matrix *H){
114     //Hadamard gate
115     H->ncol = 2;
116     H->nrows = 2;
117     H->vector = (double*) calloc(H->nrows * H->ncol, sizeof(double));
118     H->vector[0] = H->vector[1] = H->vector[2] = 1/sqrt(2);
119     H->vector[3] = -1/sqrt(2);
120     return;
121 }
122
123 /*Receive an operator O (Matrix type) and a qbits q compute O|q>
    (qbits type)*/
```

```
124 void operator_qbits_matrix(qbits q, Matrix O, qbits *res){
125     Matrix vect_q, res_poids;
126     int i;
127     //Creating the vector (Matrix type) from the qbits
128     create_matrix(&vect_q, q.n,1);
129     for(i = 0;i<q.n;i++){
130         vect_q.vector[i] = q.poids[i];
131     }
132 
133     //Compute O|q> store it in a Matrix
134     mult_matrix(O, vect_q, &res_poids);
135 
136     //Construct the qbits from the previous result
137     create_qbits(res,q.n);
138     fill_qbits(res, res_poids.vector, q.bits, q.n);
139     return;
140 }
141 
142 /*Receive a, n and res a list of size 0 wich will store the n-th
        conver
143  * gent of a*/
144 void convergent(double a, int n, int * res){
145     printf("Debut de fonction : a = %f, n = %d\n",a,n);
146     int i,b;
147     double f;
148 
149     for(i=0; i<n; i++){
150         printf("i = %d",i);
151         b = floor(a);
152         printf("  b = %d  ",b);
153         res[i] = b;
154         f = a-b;
155         if(f==0){
156             return;
157         }
158         a = 1.0/f;
159     }
160 
161 
162     printf("\n[");
163     for(i=0; i<n;i++){
164         printf("%d ",res[i]);
165     }
166     printf("]\n");
167 
```

```
168     return;
169 }
170 
171 //Main
    ----------------------------------------------------------------------
172 int main(void){
173     //~ qbits q;
174     //~ int b[2];
175     //~ double p[2];
176 
177     //~ b[0] = 0; b[1] = 1;
178     //~ p[0] = 1; p[1] = 0;
179     //~ create_qbits(&q, 2);
180     //~ fill_qbits(&q, p, b,2);
181     //~ print_qbits(q);
182     //~ printf("Fin test qbits\n");
183 
184 
185     //~ Matrix Hadamard;
186     //~ create_logical_gate(&Hadamard);
187     //~ print_matrix(Hadamard);
188     //~ printf("Fin test matrix\n");
189 
190     //~ qbits test;
191     //~ operator_qbits_matrix(q, Hadamard, &test);
192     //~ print_qbits(test);
193     //~ printf("Fin test operators\n");
194     printf("Test convergent : \n");
195     int* conv;
196     int n;
197     n = 5;
198     conv = (int*) calloc(n,sizeof(int));
199     convergent(-329.0/1380, n, conv);
200     int i;
201     printf("[");
202     for(i=0; i<n;i++){
203         printf("%d ",conv[i]);
204     }
205     printf("]\n");
206 
207     return 0;
208 }
```

quantique.c

Bibliography

[1] R. Coolman, *What Is Quantum Mechanics?*. Article, Live Science, 2014. Available at: `https://www.livescience.com/33816-quantum-mechanics-explanation.html`

[2] R.P. Feynman, *Simulating Physics with Computers*, International Journal of Theoretical Physics, vol. 21, no. 6/7, pp. 467–488, 1982.

[3] S. Lang, *Introduction to Diphantine Approximations*, chap. 1. Springer-Verlag, New York, 2nd edition, 1995.

[4] F. Xi Lin, *Shor's Algorithm and the Quantum Fourier Transform*. McGill University, 2013.

[5] J. von Neumann, *Mathematical Foundations of Quantum Mechanics*, 1932.

[6] M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2000.

[7] D. Petritis, *Quantum Mechanics: Foundations and Applications*. Lecture Notes, Université de Rennes 1, 2018. Available at: `https://perso.univ-rennes1.fr/dimitri.petritis/enseignement/mq/iq.pdf`

[8] J. Preskill, *Quantum Information*. Lecture Notes, California Institute of Technology, 2015. Available at: http://www.theory.caltech.edu/people/preskill/ph229/

[9] R.L. Rivest, A. Shamir and L. Adleman, "*A Method for Obtaining Digital Signatures and Public-key Cryptosystems*, Communications of the ACM, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[10] P. W. Shor. *Algorithm for Quantum Computation: Discrete Logarithms and Factoring*. Symposium on Foundations of Computer Science, 1994.

[11] P.W. Shor, *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, SIAM Journal on Computing, no. 5, p. 1484, 1997.