CentraleSupélec

Applications of Quantum Calculus: Implementation of Shor's Algorithm

Long Project 2nd year

Students:
Ahmed BEN AISSA
Elie MOKBEL
Henrique MIYAMOTO
Pierre MINSSEN

Supervisor:
Prof. Benoît VALIRON

February 20, 2019

Contents

1 Basic concepts **2**

2 Shor's algorithm **6**

3 Oracle circuit **12**

4 Implementation **16**

5 Complexity Analysis **17**

1 Basic concepts

Classical computers operate on strings of bits (0 or 1) and produce other strings of bits. Classical data is supposed to be clonable, erasable, readable and not supposed to change when left untouched.

In quantum computation, on the other hand, the bits are replaced by *quantum bits* or *qubits*, which are unitary elements of the 2-dimensional complex Hilbert space $\mathbb{C}^2$. We choose the orthonormal basis called *computational basis*

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

A general qubit can be seen a superposition of states $|0\rangle$ and $|1\rangle$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where $\alpha, \beta \in \mathbb{C}$.

The Hilbert space $\mathbb{C}^2$ is provided with an *inner product* $\langle\varphi|\psi\rangle = |\varphi\rangle^\dagger|\psi\rangle = \sum_i \overline{\varphi}_i\psi_i$, which allows one to the define the *norm* of a state $\||\psi\rangle\| = \sqrt{\langle\psi|\psi\rangle}$ and *orthogonality* between two states when $\langle\varphi|\psi\rangle = 0$.

Bloch spehre. The Bloch sphere is a representation of quantum states on S^2. Let us consider the qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and the polar representations $\alpha = |\alpha|e^{i\gamma} = \cos\frac{\theta}{2}e^{i\gamma}$ and $\beta = |\beta|e^{i(\gamma+\varphi)} = \sin\frac{\theta}{2}e^{i(\gamma+\varphi)}$. Then, we can write $|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right)$. Neglecting the global phase factor $e^{i\gamma}$[1], we have the mapping

$$(\theta, \varphi) \mapsto \left(\cos\frac{\theta}{2},\ e^{i\varphi}\sin\frac{\theta}{2}\right),$$

with $\theta \in [0\,\pi]$ and $\varphi \in [0, 2\pi[$.

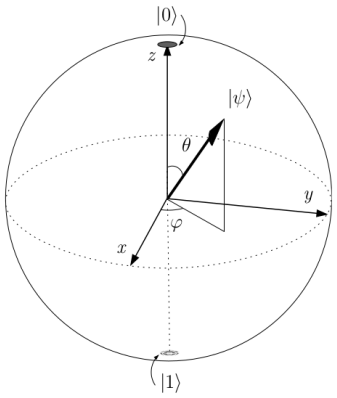


Figure 1: Bloch sphere [2].

[1]One reason for doing so is that this factor does not change the modulus squared of amplitudes $|\alpha|^2$ and $|\beta|^2$ [2].

Measurements. It is a probabilistic operation that allows one to recover some classical information. The measurement of $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ returns $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. It also alters the state of a qubit and forces it to collapse to state $|0\rangle$ or $|1\rangle$, respectively. In this case, we say the measurement was done against the computational basis $\{|0\rangle, |1\rangle\}$.

Unitary operations. The temporal evolution of an isolated quantum system is described by linear transformations, represented by matrices. Transformations that map unitary vector onto unitary vectors are called *unitary transformations* U and can be defined by the following property:

$$U^\dagger U = UU^\dagger = I,$$

where $U^\dagger = \left(\overline{U}\right)^T$ is the adjoint matrix and I is the identity. These are reversible operations.

Some usual gates are NOT, Hadamard, phase-shift and phase-flip:

- NOT
$$N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
- Hadamard
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$
- Phase-shift
$$V_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$$
- Phase-flip[2]
$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Tensor product. The tensor product between two states

$$|\psi\rangle = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_m \end{pmatrix} \text{ and } |\varphi\rangle = \begin{pmatrix} \varphi_1 \\ \vdots \\ \varphi_p \end{pmatrix}$$

is computed as

$$|\psi\rangle \otimes |\varphi\rangle = \begin{pmatrix} \psi_1\varphi_1 \\ \vdots \\ \psi_1\varphi_p \\ \psi_2\varphi_1 \\ \vdots \\ \psi_2\varphi_p \\ \vdots \\ \psi_m\varphi_1 \\ \vdots \\ \psi_m\varphi_p \end{pmatrix}.$$

[2]Note that, in fact, $Z = V_\pi$.

In general, given two matrices $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{p\times q}$, the tensor product is the matrix $A \otimes B \in \mathbb{C}^{mp\times nq}$ given by

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}$$

where a_{ij} is the (i,j)-element of A.

Given two linear transformations A and B, we can define a new linear mapping by

$$(A \otimes B)(|u\rangle \otimes |v\rangle) = A|u\rangle \otimes B|v\rangle.$$

Notation:

- We can indistinguishably write $|\psi\rangle \otimes |\varphi\rangle = |\psi\rangle|\varphi\rangle = |\psi, \varphi\rangle = |\psi\varphi\rangle$.
- $|\psi\rangle^{\otimes n} = \underbrace{|\psi\rangle \otimes \cdots |\psi\rangle}_{n \text{ times}}$ and $A^{\otimes n} = \underbrace{A \otimes \cdots A}_{n \text{ times}}$.

Two or more qubits systems. The state of a 2-qubit is an element of the tensor product space $\mathbb{C}^4 = \mathbb{C}^2 \otimes \mathbb{C}^2$, which is spanned by

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |10\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

A generic state of 2 qubits it therefore of the form

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

with $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$.

In general, writing states as the decimal number correspoding to the binary representation (e.g. $|11\rangle \to |3\rangle$), a n-qubit state is described as

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle, \text{ with } \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1.$$

As a generalisation of the 1-qubit case, the measurement of a n-qubit state changes it and forces it to collapse to one of the possible $|i\rangle$ states, each of which is measured with probability $|\alpha_i|^2$.

Among unitary operations available to 2-qubits states, we have the swap gate X and the control-not gate N_C:

$$X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad N_C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

The control-not gate changes the state of the second qubit only if the first qubit is in the state $|1\rangle$. It implements the mapping $|xy\rangle \mapsto |x\rangle \otimes |x \oplus y\rangle$.

The Toffoli gate acts on 3 qubits and is a "control-control-not": it implements the function $|xyz\rangle \mapsto |xy\rangle \otimes |z \oplus xy\rangle$, i.e., it changes the state of the last qubit if the two first qubits are in state $|11\rangle$.

Quantum entanglement. Consider the states $|\psi\rangle = a|0\rangle + b|1\rangle$ and $|\varphi\rangle = c|0\rangle + d|1\rangle$. Their tensor product is $|\psi\varphi\rangle = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$. It turns out that a general state is not on this form, unless it has $\alpha\delta = \beta\gamma$.

We say a quantum state is *entangled* when it cannot be written as a tensor product of two other states. For example, $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ cannot have such a decomposition, and hence is an entangled state.

Quantum circuits Quantum circuits are graphical representations of a procedure, i.e., a sequence of logical operations performed on a system. Unlike classical circuits, the wires must not be regarded as physical connections and their components are not available "on the shelf".

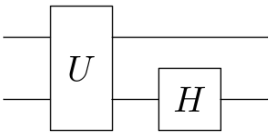


Figure 2: Example of quantum circuit representing $|\psi\rangle \mapsto (I \otimes H)(U|\psi\rangle)$ [1].

2 Shor's algorithm

The advantage of quantum algorithms over classical ones appears when using some quantum property such as entanglement or the interference, brought by complex coefficients.

In fact, quantum algorithms are based on a few "real" quantum constructions, such as quantum Fourier transform, quantum walk and amplitude amplification. The rest is composed of classical analysis and possibly an *oracle*: a quantum circuit corresponding to a reversible operation.

Some interesting problems in algebra and number theory reduce to the problem of order finding. For example, the problem of factorising an integer number, which is addressed by Shor's algorithm [4].

Factorisation. The objective of the *factorisation problem* is to factorise a big number N into prime numbers. Note that at least $n = \lceil \log_2 N \rceil$ qubits are needed to store N and that n is the maximum number of prime factors. We will show how this problem reduces to the problem of finding the order of a randomly generated integer $x < N$ generated randomly.

If N is even, 2 is trivially a factor. In addition, if x and N have common factors, then $\gcd(x, N)$ gives a factor of N; so we focus on investigating the case when x and N are coprimes.

Definition 2.1. *The order of x modulo N is the least positive integer r such that*

$$x^r \equiv 1 \mod N.$$

Theorem 2.1 (Euler's Theorem)**.** *If x and N are coprime positive integers, then*

$$x^{\varphi(N)} \equiv 1 \mod N,$$

where $\varphi(N)$ is the Euler's totient function, i.e., it indicates the number of coprimes to N which are less or equal to it.

The *order finding problem* is to find r, given x and N coprimes. The algorithm is built up on the following two theorems.

Theorem 2.2. *Let N be a composite number stored with n qubits and x be non-trivial solution of the equation $x^2 \equiv 1 \mod N$ in the range $1 \leq x \leq N$, i.e., $x \not\equiv \pm 1 \mod N$. Then at least one of $\gcd(x+1, N)$ and $\gcd(x-1, N)$ is a non-trivial factor of N.*

Proof. Note that $x^2 \equiv 1 \mod N \Leftrightarrow x^2 - 1 \equiv 0 \mod N \Leftrightarrow (x+1)(x-1) \equiv 0 \mod N$, which means that N is a divisor of $(x+1)(x-1)$. If $1 < x < N-1$, then $0 < x-1 < x+1 < N$ and N cannot be a divisor of $(x+1)$ neither of $(x-1)$ separately. So both $(x+1)$ and $(x-1)$ must have factors of N. In this case, at least one of $\gcd(x+1, N)$ and $\gcd(x-1, N)$ produce a non-trivial factor of N[3]. □

Theorem 2.3. *Suppose $N = p_1^{\alpha_1} ... p_m^{\alpha_m}$ is the prime factorisation of and odd composite positive integer. Let x be an integer chose uniformly at random such that $1 \leq x \leq N-1$ and $\gcd(x, N) = 1$. Let r be the order of x modulo N. Then*

$$\Pr\{r \text{ is even and } x^{r/2} \not\equiv \pm 1 \mod N\} \geq 1 - \frac{1}{2^m}.$$

[3]Such factor can be computed by using Euclid's algorithm.

Proof. See [5], pp. 751-752[4]. □

The algorithm runs as follows.

Algorithm 1: Shor's algorithm for factorising N.

1 If N is even, return the factor 2.
2 Randomly choose x such that $1 \leq x \leq N-1$.
3 If $\gcd(x, N) \neq 1$, return the factor $\gcd(x, N)$.
4 Find the order r of x modulo N.
5 If r is even and $x^{r/2} \not\equiv \pm 1 \mod N$, then $\gcd(x^{r/2}+1, N)$ and $\gcd(x^{r/2}-1, N)$ are non-trivial factors.
6 Else, restart the algorithm.

Some remarks:

- The method fails if N is the power of a prime number. But in this case there exists a a classical algorithm to solve the problem [2].
- The problem is finding the order of x modulo N, for which there is no efficient classical procedure available. This problem is addressed in the next part.

Order finding. The problem of order finding, i.e., given x and N coprimes, to find r such that $x^r \equiv 1 \mod N$ is related to the matrix eigendecomposition. A unitary U, being a Hermitian matrix, can be decomposed as

$$U = \sum_j \lambda_j u_j u_j^\dagger,$$

where u_j are orthonormal eigenvectors and λ_j are the associated eigenvalues.

Let us assume that $N = 2^n$ and consider the operation U_x on n qubits that implements the mapping

$$U_x : \quad |j\rangle \mapsto |j \cdot x \mod N\rangle.$$

The operator is unitary because x and N are coprimes and the image of $\{0, ..., N-1\}$ is the whole set. In particular, U_x^k sends $|j\rangle \mapsto |j \cdot x^k \mod N\rangle$, so if $x^r \equiv 1 \mod N$, the map U_x^r is the identity map.

The eigenstates of U are of the [5]

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i s k}{r}\right) |x^k \mod N\rangle$$

for $0 \leq s \leq r-1$. The eigenvalues are the r-th roots of the unity, having the form $e^{2\pi i s/r}$, since

$$U|u_s\rangle = \exp\left(\frac{2\pi i s}{r}\right)|u_s\rangle.$$

An algorithm for finding such eigenvalues should be enough to find the order r of x modulo N[6].

[4] Actually, these authors use a slightly different bound for the probability: $1 - 1/2^{m-1}$.
[5] To simplify notation, hereafter we shall denote U_x by U simply.
[6] An alternative unitary could have been used: $V|j\rangle|k\rangle = |j\rangle|k + x^j \mod N\rangle$.

Quantum Fourier transform. The Fourier transform is an operation that can be performed faster in quantum computers. The quantum Fourier transform (QFT) [6] is defined in analogy with the discrete Fourier transform. It is the linear mapping:

$$\begin{aligned} \hat{f} \ : \ & \mathbb{C}^N \to \mathbb{C}^N \\ & |j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k/N} |k\rangle \end{aligned}$$

Let us consider $N = 2^n$ with orthonormal basis $\{|0\rangle, \ldots, |2^n - 1\rangle\}$. We shall use the *binary representation* $j =: j_1 j_2 \ldots j_n$ to represent $j = \sum_{i=1}^{n} j_i 2^{n-i}$ and the *binary fraction* $j =: 0.j_1 j_2 \ldots j_n$ to represent $j = \sum_{i=1}^{n} j_i 2^{-i}$.

It will be useful to consider an alternative form of the QFT which is indeed so important that could be considered its definition itself:

$$|j_1 \ldots j_n\rangle \mapsto \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i 0.j_1 \ldots j_n}|1\rangle\right)}{2^{n/2}}.$$

The equivalence between the two expressions can be shown as follows [3]:

$$\begin{aligned} & \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n} |k\rangle \\ = & \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j (\sum_{l=1}^{n} k_l/2^l)} |k_1 \ldots k_n\rangle && \text{(write in binary representation)} \\ = & \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \prod_{l=1}^{n} \otimes e^{2\pi i j k_l/2^l} |k_l\rangle && \text{(tensor product decomposition)} \\ = & \frac{1}{2^{n/2}} \prod_{l=1}^{n} \otimes \left(\sum_{k_l=0}^{1} e^{2\pi i j k_l/2^l} |k_l\rangle \right) && \text{(factorise the binary powers)} \\ = & \frac{1}{2^{n/2}} \prod_{l=1}^{n} \otimes \left(|0\rangle + e^{2\pi i j/2^l} |1\rangle \right). \end{aligned}$$

The product representation allows one to implement the quantum circuit for the QFT using Hadamard and R_k rotation gates, the latter being of the form[7]

$$R_k := \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}.$$

To conclude this section, we remark that the QFT, as a linear operation, may be written in matrix form[7]:

[7]The attentive reader will notice that $R_k = V_{2\pi/2^k}$.

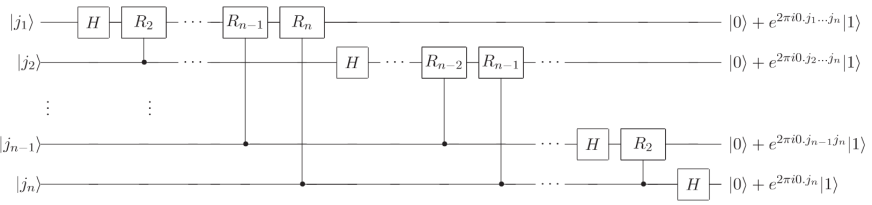


Figure 3: Quantum circuit that implements the QFT [3].

$$F = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \omega_n^3 & \cdots & \omega_n^{N-1} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \cdots & \omega_n^{2(N-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \cdots & \omega_n^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{N-1} & \omega_n^{2(N-1)} & \omega_n^{3(N-1)} & \cdots & \omega_n^{(N-1)(N-1)} \end{pmatrix}$$

where $N = 2^n$ and $\omega_n := e^{2\pi i/2^n}$. This allows one to verify that $FF^\dagger = F^\dagger F = I$, therefore concluding that the Fourier transform is a unitary transformation.

Quantum phase estimation. Suppose a unitary operator U has an eigenvector $|u\rangle$ with eigenvalue $e^{2\pi i\varphi}$, where φ is unknown. The objective of the *phase estimation subroutine* is to estimate the value of φ. We assume we have oracles capable of preparing the state $|u\rangle$ and performing controlled-U^{2^j} operations.

The procedure uses two registers: the first contains t qubits initially in state $|0\rangle$. The number t depends on the desired accuracy for φ and on the probability we want the procedure to be successful. The second register begins with the state $|u\rangle$ and contains as many qubits as necessary to store it.

The first step is to apply the following circuit to the registers.

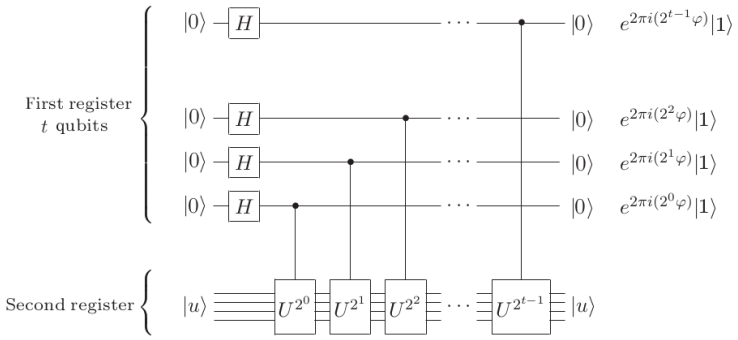


Figure 4: Quantum circuit for first step of phase estimation (normalisation has been omitted on the right side) [3].

The final state of the first register is

$$\frac{1}{2^{t/2}}\left(|0\rangle + e^{2\pi i 2^{t-1}\varphi}|1\rangle\right)\left(|0\rangle + e^{2\pi i 2^{t-2}\varphi}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i 2^{0}\varphi}|1\rangle\right)$$
$$= \frac{1}{2^{t/2}}\sum_{k=0}^{2^t-1} e^{2\pi i\varphi k}|k\rangle.$$

The second step is to apply the inverse Fourier transform on the first register[8] and the third step is to measure it. Note that, if we write the previous expression using the binary fraction representation, we have

$$\frac{1}{2^{t/2}}\left(|0\rangle + e^{2\pi i 0.\varphi_t}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.\varphi_{t-1}\varphi_t}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i 0.\varphi_1\ldots\varphi_t}|1\rangle\right).$$

Comparing it with the (product) expression for the Fourier transform, one can see that the result of applying the inverse Fourier transform is the state $|\varphi_1\ldots\varphi_r\rangle$ and a measurement in the computational basis gives exactly an estimation for φ!

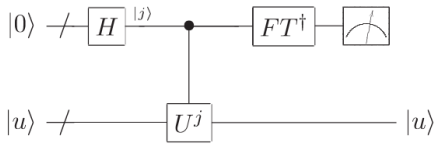


Figure 5: Schematic of the overall phase estimation subroutine [3] .

Summarising, the phase estimation subroutine gives an estimation $\tilde{\varphi}$ to the phase of an eigenvalue of a unitary U, which is precisely what we wanted to implement the order finding algorithm. The inverse Fourier transform performs

$$\frac{1}{2^{t/2}}\sum_{j=0}^{2^{t-1}} e^{2\pi i\varphi j}|j\rangle|u\rangle \mapsto |\tilde{\varphi}\rangle|u\rangle.$$

Theorem 2.4 (Accuracy of φ [3], pp. 223-224)**.** *To successfully obtain φ accurate to n bits with probability of success at least $1-\epsilon$, we choose*

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

Continued fraction expansion. Let us recapitulate what we have so far: we have reduced the factorisation problem to the order finding problem, which can be solved by calculating eigenvalues of a unitary U of the form $e^{2\pi i s/r}$. The quantum phase estimation procedure uses the QFT to estimate the phase φ of an eigenvalue $e^{2\pi i\varphi}$. So we have an estimation

$$\tilde{\varphi} \approx \frac{s}{r}.$$

It misses one building block in order to solve our problem: how to retrieve r from $\tilde{\varphi}$. To do so, we shall use *continued fraction expansions*. The results on this topic can be deepened in [8].

[8] To do so, we just have to invert the circuit for QFT on Figure 3.

Definition 2.2. *A finite simple continued fraction is an expression of the form*

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

which is denoted by $[a_0, a_1, a_2, \cdots, a_N]$*, with* $a_i \in \mathbb{N}^* \ \forall i \in [\![1, N]\!]$.
We define the n-th convergent to this continued fraction as $[a_0, \ldots, a_n]$ *for* $0 \leq n \leq N$.

Theorem 2.5. *The n-th convergent may be written as a fraction*

$$[a_0, \ldots, a_n] = \frac{p_n}{q_n}$$

whose coefficients are given by the following recurrence relation:

$$\begin{array}{lll} p_0 := a_0 & p_1 := a_1 a_0 + 1 & p_n := a_n p_{n-1} + p_{n-2} \\ q_0 := 1 & q_1 := a & q_n := a_n q_{n-1} + q_{n-2} \end{array} \quad (2 \leq n \leq N).$$

Proof. See [8], pp. 166-167. □

The next two theorems will present the *continued fractions algorithm* to approximate $\tilde{\varphi}$ by a n-th convergent and explain why it suffices to solve our problem.

Theorem 2.6 (Continued fraction algorithm)**.** *Any rational number x can be represented by a finite simple continued fraction* $[a_0, \ldots, a_N]$.

Proof. See [8], pp. 173-174. □

The algorithm runs as follows.

Algorithm 2: Continued fraction algorithm for rational x.

1 Set $a_0 = \lfloor x \rfloor$. Then $x = a_0 + \xi_0$, with $\xi_0 \in [0, 1[$.
2 While $\xi_i \neq 0$: $a_{i+1} = \lfloor 1/\xi_i \rfloor$ and $1/\xi_i = a_{i+1} + \xi_{i+1}$, with $\xi_{i+1} \in [0, 1[$.
3 The continued fraction is $[a_0, \ldots, a_N]$, where N is such that $\xi_N = 0$.

Theorem 2.7. *Suppose* s/r *is a rational number such that*

$$\left| \frac{s}{r} - \varphi \right| \leq \frac{1}{2r^2}.$$

Then s/r *is a convergent of the continued fraction for* φ.

Proof. See [8], pp. 196-197. □

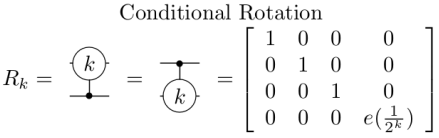
Since $\tilde{\varphi}$ is an approximation of s/r accurate to $n = \lceil \log_2 N \rceil$ qubits, the theorem applies. Summarising, given $\tilde{\varphi}$, we can use the continued fraction algorithm to find an irreducible fraction $s'/r' = s/r$. Then, r' is our candidate for the order. We check calculating $x^{r'} \mod N$; if the result is 1, we are done!

3 Oracle circuit

Now that we know the quantum algorithm to factorise a number, we can devote our attention to the implementation in terms of quantum circuit. Although we have already provided a circuit for the QFT, the oracle remains a mysterious black box. In this Section, we show how it can be implemented. In [9], the author presents a circuit implementation that aims to reduce the number of qubits needed, while using a polynomial number of elementary quantum gates.

Adder gate. An initial version for the adder circuit is presented by [10]. Instead of basing on classical circuits which use at least $3n$ qubits to sum two n-qubits number, by composing carry gates and sum gates, the author presents an implementation based on the QFT. The idea, in order to sum a and b, is to compute the Quantum Fourier transform of a, $\phi(a)$, and then use b to compute $\phi(a+b)$. The inverse QFT is then applied to recover the desired result.

Using the following notation[9] to conditional rotation gates R_k, it is possible to describe the adder circuit as in Figure 6.



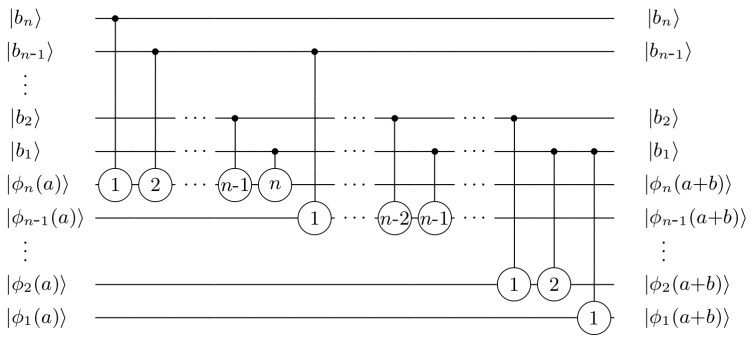


Figure 6: Quantum adder based on QFT, as presented in [10]. The QFT operator is denoted $\phi(\cdot)$.

The advantage of this method is that a quantum computer that can perform simultaneous calculations can decrease the time of execution, for instance, by executing all depth 1 rotations simultaneously, and then all depth 2 rotations and so on. In particular, in the case of using the Approximate Quantum Fourier Transform (AQFT)[10], this quantum addition can be computed in $n \log_2 n$ op-

[9] Here, understand $e\left(\frac{1}{2^k}\right)$ as $\exp\left(\frac{2\pi i}{2^k}\right)$.

[10] This technique consists in reducing the rotations blocks as k gets large, as for large values of k, the rotation block approaches an identity gate ($R_k \approx I$). The optimal value of k is around $\log_2 n$ [10].

erations.

To follow what happens in this circuit, we shall resume the notation introduced in the presentation of the QFT (§2).

$$
\begin{aligned}
|\phi_n(a)\rangle &\to \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.a_n a_{n-1}\cdots a_1 + 0.b_n)}|1\rangle\right) && (R_1 \text{ rotation from } b_n)\\
&\to \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.a_n a_{n-1}\cdots a_1 + 0.b_n b_{n-1})}|1\rangle\right) && (R_2 \text{ rotation from } b_{n-1})\\
&\;\;\vdots\\
&\to \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.a_n a_{n-1}\cdots a_1 + 0.b_n b_{n-1}\cdots b_1)}|1\rangle\right) && (R_n \text{ rotation from } b_1)\\
&= |\phi(a+b)\rangle.
\end{aligned}
$$

A slight variation of the implementation presented so far is introduced by [9], with the notation ϕ-ADD(a). As he points out, the objective of implementing this sum is to afterwards be able to retrieve the order modulo N of a (which plays the role of x in the previous notation). Thus, a is classical and the qubits that represent it may be replaced by classical bits. An additional qubit is added to prevent overflow.

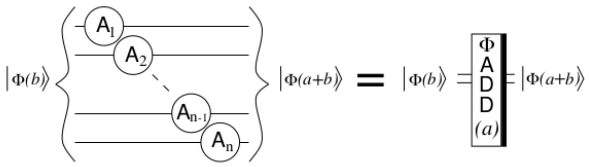


Figure 7: Circuit of ϕ-adder as presented in [9].

The reverse of this gate (denoted with the thick bar on the left) will be used to subtraction and comparison.

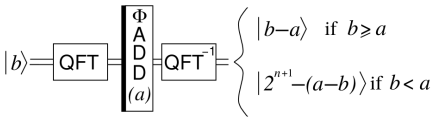


Figure 8: Circuit of reverse ϕ-ADD(a) [9].

Modular adder gate. The goal now is to use the ϕ-ADD(a) to build a adder modulo N. This means that we have to subtract N from the result if $a+b \geq N$. We will denote this block ϕ-ADD(a)-mod-N. The whole block is controlled by two qubits.

The circuit operation, depicted in Figure 9 is described as follows:

- The circuit begins with $|\phi(b)\rangle$ as input.

- After the ϕ-ADD(a) block, the value in the register is $\phi(a+b)$ (with no overflow, assuming there is an extra $|0\rangle$ qubit in $|b\rangle$).
- Then, we apply a reverse ϕ-ADDN, having $\phi(a+b-N)$ on the register. We apply this operation even without knowing if it was really necessary to subtract N from the previous result.
- Now we have to check if this operation was indeed necessary or not. But to access the most significant bit of it, we have to pass the whole value through an inverse QFT gate. The most significant bit is used to control a CNOT gate on the ancilla qubit $|0\rangle$.
- The QFT is reapplied, so we come back to $\phi(a+b-N)$.
- The value N is added back if the subtraction was unnecessary, so that $a+b<N$. The value in the register is now $\phi(a+b) \mod N$.
- Close to being satisfied, we just have to restore the ancilla to assure that our gate corresponds to a reversible operation. The trick is to note that $(a+b) \mod N \Leftrightarrow a+b<N$. So one can apply a circuit analogous to the previous one: subtract a, recover the most significant qubit of the result (which will be $|0\rangle$), invert it and use it on controlled-not over the ancilla, which will be restored to $|0\rangle$; at the end, restore a.

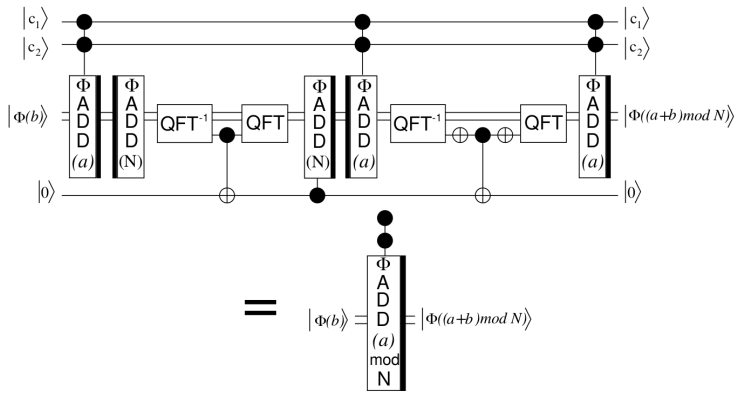


Figure 9: Circuit of ϕ-adder modulo N as presented in [9].

Controlled multiplier gate. Now, we can use the previous ϕ-ADD(a)-mod-N to build a controlled multiplied gate, denoted CMULT(a)-mod-N. This gate has input $|c\rangle|x\rangle|b\rangle$ and is to compute $|c\rangle|x\rangle|(b+ax) \mod N\rangle$ if $|c\rangle = |1\rangle$, or not to alter any qubit otherwise.

The implementation with ϕ-ADD(a)-mod-N is based on the fact that

$$(ax) \mod N = \sum_{i=0}^{n-1} 2^i a x_i \mod N$$

where the modulo N has to be applied on each step of the summation. Thus all we need are n successive ϕ-ADD($2^i a$)-mod-N gates.

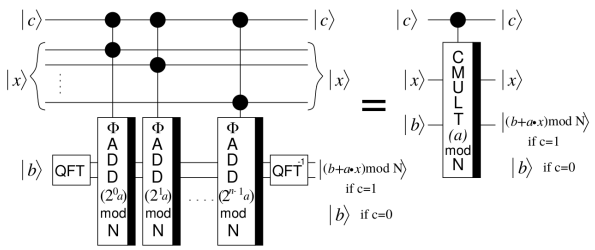


Figure 10: Circuit of CMULT(a)-mod-N as presented in [9].

Controlled U gate. In the previous step, we got a controlled gate that implements the mapping $|x\rangle|b\rangle \mapsto |x\rangle|b + (ax) \mod N\rangle$. However, in fact, we would like to implement $|x\rangle \mapsto |x\rangle|(ax) \mod N\rangle$, which corresponds to our U-gate unitary operation. The implementation is presented in Figure 11.

Let us follow the steps of the circuit:

- Start by applying the CMULT(a)-mod-N gate to the input $|c\rangle|x\rangle|0\rangle$.
- Apply a controlled-SWAP gate (controlled by $|c\rangle$ as well). Indeed, it is only necessary to apply it on n qubits, because the most significant one of $(ax) \mod N$ will always be $|0\rangle$ (to store the overflow of the ϕ-adder).
- Then, apply the inverse CMULT(a^{-1})-mod-N[11].

The sequence of performed operations is

$$|x\rangle|0\rangle \to |x\rangle|(ax) \mod N\rangle \to |(ax) \mod N\rangle|x\rangle \to$$
$$|(ax) \mod N\rangle|x - a^{-1}ax\rangle = |(ax) \mod N\rangle|0\rangle.$$

As a result, the controlled-U gate implements $|x\rangle|0\rangle \mapsto |(ax) \mod N\rangle|0\rangle$. To compute a $(\text{control} - U_a)^n$, it is enough to apply $\text{control} - U_{a^n}$ instead of applying the same gate multiple times.

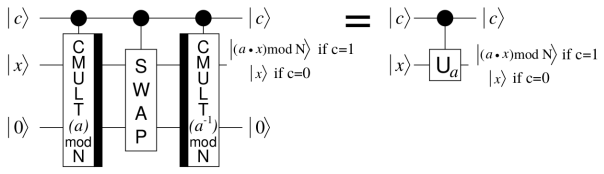


Figure 11: Circuit of controlled U-gate as presented in [9].

[11] The inverse modulo N of a, a^{-1}, can be computed via Euclid's algorithm.

4 Implementation

The implementations was made using *ProjectQ* [11, 12]. In fact, they provide an example code for Shor's algorithm[12].

[12]Available in: `https://github.com/ProjectQ-Framework/ProjectQ/blob/develop/examples/shor.py`

5 Complexity Analysis

A complexity analysis is performed by [9], presenting number of qubits, order of number of gates and order of depth for each part of the circuit proposed by them (cf. §3) to factorise an integer N stored with $n = \lceil \log_2 N \rceil$ bits. The results are summarised in Table 1.

Table 1: Complexity analysis of circuit parts [9].

Circuit	Number of qubits	Order of gates	Order of depth
ϕ-ADD(a)	$n+1$	$O(n)$	$O(1)$
ϕ-ADD(a)-mod-N	$n+4$	$O(nk_{\max})$	$O(n)$
CMULT(a)-mod-N	$2n+3$	$O(n^2 k_{\max})$	$O(n^2)$
U_a gate	$2n+3$	$O(n^2 k_{\max})$	$O(n^2)$
Shor	$2n+3$	$O(n^3 k_{\max})$	$O(n^3)$

When using the exact QFT, $k_{\max} = n$.

We have compared the number of gates of some of the circuits we have implemented with the result proposed above. Using `MATLAB`, we fitted each set of data to a polynomial function of the respective degree (Table 2, Figures 12, 13, 14). Note that, having used used the exact QFT approach in our implementation, $k_{\max} = n$[13]. We have performed tests for $N = 2^k - 1, \quad k \in [\![3, 19]\!]$ and we counted the swap gates X used for qubits initialisation in the total number of gates.

Table 2: Order of number of gates and fitting polynomials in n.

Circuit	Order	Fitting polynomial
CMULT(a)-mod-N	$O(n^3)$	$2.5\ n^3 + 7.5\ n^2 + 18\ n + 11$
U_a gate	$O(n^3)$	$5\ n^3 + 15\ n^2 + 32\ n + 12$
Shor	$O(n^4)$	$10\ n^4 + 26\ n^2 - 16\ n + 2$

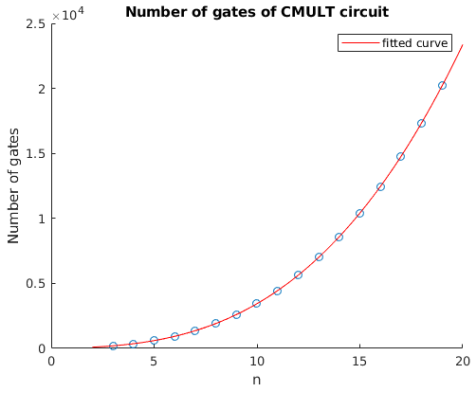


Figure 12: Number of gates of CMULT(a)-mod-N circuit as function of n.

[13] This relation can be reduced with the approximate QFT technique up to $k_{\max} = O(\log(n/\epsilon))$, for any ϵ polynomial in $1/n$ [9].

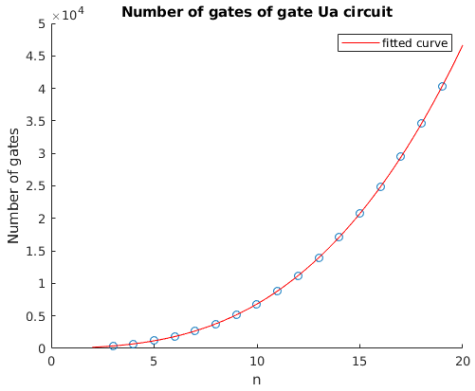


Figure 13: Number of gates of U_a gate circuit as function of n.

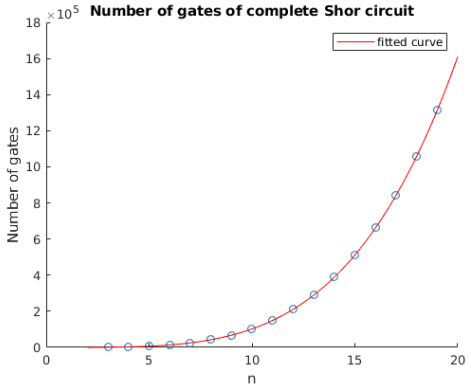


Figure 14: Number of gates of Shor's algorithm circuit as function of n.

We observe that all resulting polynomials fit well the sets of data, matching the theoretical results in Table 1.

References

[1] Valiron, B. "Quantum computation: a tutorial". *New Generation Computing*, vol. 30, no. 4, pp. 271-296, oct. 2012.

[2] Portugal, R. et al. *Uma introdução à computação quântica*. São Carlos: SBMAC, 2004.

[3] Nielsen, Michael A. and Chuang, Isaac L. *Quantum computation and quantum information*. 10th anniversary edition. Cambridge: Cambridge University Press, 2010.

[4] Shor, Peter. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484-1509, 1997.

[5] Ekert, A. and Jozsa, R. "Quantum computation and Shor's factoring algorithm". *Reviews of Modern Physics*, vol. 68, no. 3, pp. 733-753, 1996.

[6] Griffiths, Robert B. and Niu, Chi-Sheng. "Semiclassical Fourier Transform for Quantum Computation". *Physycal Review Letters*, vol. 76, no. 17, pp. 3228-3231, 1996.

[7] Wikipedia. *Quantum Fourier transform*. Available in: `https://en.wikipedia.org/wiki/Quantum_Fourier_transform`. Access on: 18 dec. 2018.

[8] Hardy, G. H. and E. M. Wright. *An introduction to the theory of numbers*. 6th edition. Oxford : Oxford University Press, 2008.

[9] Beauregard, Stephane. "Circuit for Shor's algorithm using $2n + 3$ qubits". *Quantum Information and Computation*, vol. 3, no. 2 pp. 175-185, 2003.

[10] Draper, Thomas G. *Addition on a quantum computer*. Available in: `https://arxiv.org/abs/quant-ph/0008033`. Access on: 14 dec. 2018.

[11] ProjectQ. Availabe in: `https://projectq.ch/`. Access on: 28 nov. 2018.

[12] Steiger, Damian S.; Häner, Thomas and Troyer, Matthias. "ProjectQ: an open source software framework for quantum computing". *Quantum*, vol. 2, p. 49, jan. 2018.