Study of Shor's factoring algorithm using IBMs quantum computers

Author: Berta Casas Font
Facultat de Física, Universitat de Barcelona, Diagonal 645, 08028 Barcelona, Spain.

Advisor: Bruno Juliá Díaz

Abstract: We study Shor's algorithm for number factorization using quantum systems. The fundamental parts of this algorithm, quantum Fourier transform and phase estimation, are presented in this work. To gain insight into the key elements of the algorithm we have first implemented it using classical techniques. Afterwards, we have implemented the quantum version using IBMs qiskit language. We have tested the performance of the algorithm both on a simulator and on real quantum computers for a case study of factoring $N = 15$.

I. INTRODUCTION

Shor proposed an algorithm which has become one of the most important in the field [1]. This describes a way of solving a well-known problem in a quantum computer (QC): integer factorization. The problem is so hard to solve that even RSA public key cryptosystem (the widely used system for secure data transmission) is based on the lack of methods to solve the number factoring in polynomial time, i.e. such that the time to solve the problem scales polynomially on the size of the number to be factored. Shor's algorithm is able to factor in polynomial time in a QC. This is one of the reasons why a great amount of effort has been made to build such quantum devices.

In this work we analyse the fundamental parts of the algorithm and implement it on the IBMq devices [3]. In section II, we describe the main concepts for understanding the algorithm, such as the discrete Fourier transform (DFT) and its quantum generalization, the quantum Fourier transform (QFT) and one of the most important applications of QFT: phase estimation. Then in section III we present the structure of the algorithm and discuss its main features. We begin with an illustrative classic case and, finally, we perform the quantum implementation on an IBM quantum device for $N = 15$. Then, in section V we present the main conclusions we obtain based on the results.

II. THEORETICAL BACKGROUND

In this section we briefly describe the fundamental elements entering in Shor's algorithm.

A. Discrete Fourier Transform

Given a set of N complex numbers, $x_0, x_1, ..., x_{N-1}$, we can perform the DFT and obtain another set of N numbers $y_0, y_1, ..., y_{N-1}$, also complex, defined by

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-2\pi i jk/N} x_j \, . \tag{1}$$

B. Quantum Fourier Transform

QFT is a powerful tool for performing Fourier transforms of quantum mechanical amplitudes. This unitary operation, applied on a q qubit state and expressed in an orthonormal basis $|0\rangle, ..., |N-1\rangle$, with $N = 2^q$ can be represented as [2]

$$|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i jk/N} |k\rangle. \tag{2}$$

j can go from 0 to $N-1$ and it is a representation of a state of n qubits, $j = j_1 j_2 ... j_n$, with $j_i = \{0, 1\}$. For example, for $n = 4$, our state could be $j = |13\rangle = |1101\rangle$.

As we can observe, Eqs. (1) and (2) are formally the same, except for the sign in the exponential, which is defined like this for convention. In QFT we obtain a quantum state, instead of a set of numbers like in DFT. Hence, when we measure, we will not have access to all the amplitudes and states at the same time.

The QFT can be written more conveniently for circuit implementation as [2]

$$\begin{aligned} \frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right) \ldots \\ \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right) . \end{aligned} \tag{3}$$

Where we have applied the binary representation of an integer and a fraction which are, respectively

$$\begin{aligned} j_1 j_2 ... j_n = j = j_1 2^{n-1} + j_2 2^{n-2} + ... + j_n 2^0 \\ 0.j_1 j_2 ... j_n = j_1 2^{-1} + j_2 2^{-2} + ... + j_n 2^{-n}. \end{aligned} \tag{4}$$

An example of the QFT circuit for 3 qubits is given in Fig. 1. We have used the Hadamard gate, H, which acts on a single qubit state like $\mathrm{H}|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $\mathrm{H}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, and the controlled phase, $\mathrm{P}(\phi)$, which acts on the target qubit, only if the control qubit is set to the state $|1\rangle$, like $\mathrm{P}|0\rangle = |0\rangle$ and $\mathrm{P}|1\rangle = e^{i\phi}|1\rangle$. If we apply a P operation on a target qubit in the state $|q_t\rangle$ controlled by a qubit in a superposition of states $H|0\rangle$, we have the state $\frac{1}{\sqrt{2}}(|0\rangle \otimes |q_t\rangle + |1\rangle \otimes P|q_t\rangle)$ and now we can not treat these qubits states separately.

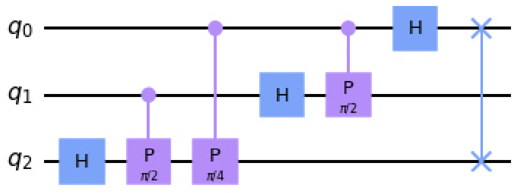

FIG. 1: Circuit implementation of the QFT for 3 qubits with IBM convention. Therefore, $j_1 = q_2$, $j_2 = q_1$ and $j_3 = q_0$. A swap gate (the last one in the figure) is needed to match with Eq. (3).

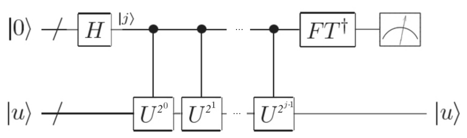

FIG. 2: Schematic circuit implementation of the phase estimation algorithm in Eq. (6). We have j qubits on the first register (first line) and the eigenstate $|u\rangle$ below. We apply one Hadamard gate to each of the first register qubits. Then we apply several powers of U, each one controlled by the first, second, until $j-1$ qubit, respectively. Finally, an inverse QFT (executed by inverting and swapping the order of the gates of the QFT circuit) is performed on the register qubits and we measure the resulting state.

C. Phase estimation

Suppose an unitary operator U such that $U|u\rangle = e^{2\pi i\varphi}|u\rangle$. The circuit which allows us to estimate the phase φ is the one in Fig. 2. The execution of U controlled by a register qubit produces a phase kickback in the above qubits. For example, if we apply U to $|u\rangle$ controlled by $H|0\rangle \propto (|0\rangle + |1\rangle)$ we have

$$|0\rangle \otimes |u\rangle + |1\rangle \otimes e^{i2\pi\varphi}|u\rangle = (|0\rangle + e^{i2\pi\varphi}|1\rangle) \otimes |u\rangle. \quad (5)$$

The phase has been transferred to the first register qubit, despite the fact that the gate was not directly applied to it.

After applying all the powers of the controlled U in Fig. 2, the remaining state is

$$\frac{1}{2^{j/2}}\left(|0\rangle + e^{2\pi i 2^{j-1}\varphi}|1\rangle\right)\left(|0\rangle + e^{2\pi i 2^{j-2}\varphi}|1\rangle\right)\dots \left(|0\rangle + e^{2\pi i 2^{0}\varphi}|1\rangle\right)\otimes|u\rangle \quad (6)$$

If we use Eq. (4) we have $\varphi = \varphi_1 2^{-1} + \varphi_2 2^{-2} + \dots + \varphi_j 2^{-j}$. Then, for the first term $e^{2\pi i 2^{j-1}\varphi} = e^{2\pi i 2^{-1}\varphi_j} = e^{2\pi i 0.\varphi_j}$, since all the other exponential are 1, as they are multiples of $2\pi i$. Following an analogous argument for the rest of register qubits states, we can write the state in Eq. (6) like

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i 0.\varphi_j}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.\varphi_{j-1}\varphi_j}|1\rangle\right)\dots \left(|0\rangle + e^{2\pi i 0.\varphi_1\varphi_2\cdots\varphi_j}|1\rangle\right) = \frac{1}{\sqrt{2^j}}\sum_{k=0}^{2^j-1} e^{2\pi i\varphi k}|k\rangle\otimes|u\rangle, \quad (7)$$

where in the last equality we have used that $\varphi < 1$ and the comparison between Eq. (3) and Eq. (2). We can see that this expression is the same in Eq. (2), except for the factor $N = 2^j$ dividing the exponential. Hence, if we perform an inverse QFT on the first register qubits we obtain the state $|2^j\varphi\rangle|u\rangle$. Measuring on this state, we get the value $2^j\varphi$ and we recover the phase φ, dividing by 2^j.

III. SHOR'S ALGORITHM

The goal of the algorithm is to find the prime factors of an integer number N, i.e. the prime numbers in which we can decompose N. It fails for even numbers, but then 2 is a trivial factor of the number. The algorithm has the following structure:

(1) Find a random number $a < N$.

(2) If $\gcd(a, N) = 1$ (the great common divisor of a and N), then we compute $f(x) = a^x \pmod N$ (the remainder of a^x divided by N). If $gcd(a, N) \neq 1$, we have found a factor of N.

(3) Find the order of the function, i.e., the period r such that $f(x + r) = f(x)$.

(4) Once we have found r, if it is odd or $a^{r/2} + 1 = 0 \pmod N$, then we have to go back to (1).

(5) Otherwise, at least one of $\gcd(a^{r/2} + 1,\ N)$ or $\gcd(a^{r/2} - 1,\ N)$ are factors of N.

Because $f(0) = a^0 = 1 \pmod N$ $\forall x$, and $\forall N$, it is equivalent to define r to be the least positive integer such that $a^r = 1 \pmod N$. Then, $a^r - 1 = 0 \pmod N$, which means that $a^r - 1 = k \cdot N$, for some integer k. Thus, N divides $a^r - 1 = (a^{r/2} + 1)(a^{r/2} - 1)$. Therefore, if r does not fall at (4), at least one of these two terms gives a factor of N when we compute the gcd with it.

A. Finding the period classically

As an example, we show how we can implement the algorithm classically, by finding the period using the DFT. For this purpose, we try to factorize $N = 765$. We choose, for example, $a = 7$. As $\gcd(765, 7) = 1$, 7 is not a factor of N and we can continue.

We compute $f(x) = 7^x \pmod{765}$. In order to find r, we have implemented a DFT, given by Eq. (1). This is shown in Fig. 3. We define $y_0 \equiv \hat{f}(\zeta = 0) =$

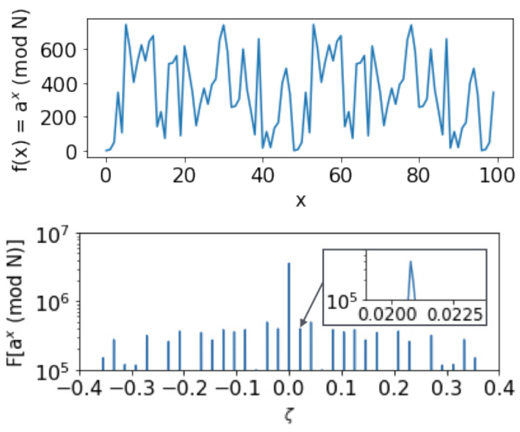

FIG. 3: Above, representation of $f(x) = 7^x \pmod{765}$ for 100 values of x. Below, semilogarithmic graph of the DFT of $f(x) = 7^x \pmod{765}$. The operation was computed with $f(x)$ between $x = 0$ and $x = 10^4$. The zoom corresponds to $\zeta_1 = 0.0208$, frequency which corresponds to a period of $T = 1/\zeta_1 \approx 49$.

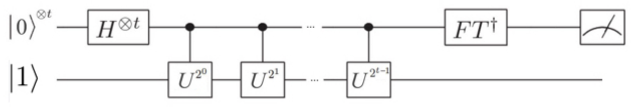

FIG. 4: Schematic procedure of the circuit implementation of Shor's Algorithm. The exponent k on U^{2^k} gives the first register qubit which controls the operation performed on the state $|1\rangle$.

$\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} f(x_j)$. Since $f(x)$ is always positive, when we compute the DFT we have a peak at $\zeta = 0$. However, we are interested in the lower frequency peak, because this gives us the larger periodicity of the function, r.

We obtain the period by computing $T = r = 1/\zeta_1 = 1/0.0208 \approx 48$. Since r is not odd and $7^{48/2}+1 \neq 0 \pmod{765}$, we can proceed with step (5). $\gcd(7^{48/2} \pm 1, 765)$ will give us at least one factor of 765. We obtain 17 and 45. These two numbers divide N: $765/17 = 45$ and $765/45 = 17$.

In conclusion, we have found two integers $p = 17$ and $q = 45$ such that $765 = p \cdot q$. However 45 is not a prime factor of 765. We would have to implement the algorithm for $N = 45$ and find its prime factors.

B. Circuit implementation

Given an integer N, the circuit needs two registers of qubits: the first register with t qubits ($N^2 \leq 2^t \leq 2N^2$) all set to zero and the second register initially in the state $|1\rangle$ (in decimal representation), with at least as many qubits as we need to represent N in binary. Later we will discuss why we need these numbers of qubits.

Following Shor's algorithm circuit in Fig. 4, we apply

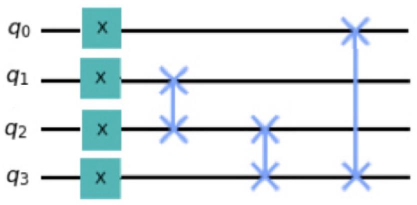

FIG. 5: Circuit implementation for the operator U that performs $U|y\rangle = |7y(\text{mod } 15)\rangle$ with $y = q_3q_2q_1q_0$ and $1 \leq y \leq 15$.

Hadamard gates to all the t first register qubits, achieving a superposition of states $\frac{1}{\sqrt{2^t}}\sum_{x=0}^{2^t-1}|x\rangle$, with x expressed in decimal basis. In some manner, this is the 'domain' of our function. After this, we use an operator U such that, given an N and an a, performs the operation

$$U|y\rangle = |ay(\text{mod N})\rangle. \tag{8}$$

For example, for $N = 15$ and $a = 7$ we have $U|1\rangle = |7 \cdot 1(\text{mod } 15)\rangle = |7\rangle$, $U^2|1\rangle = U|7\rangle = |4\rangle$ and, following this line, $U^r|1\rangle = |1\rangle$. As we can see, $|1\rangle$ is not an eigenstate of U, but it is of U^r. The circuit for the implementation of this particular U gate is given in Fig. 5.

If we apply all the other powers of the controlled U gate, we arrive at the state

$$\frac{1}{\sqrt{2^t}}\sum_{x=0}^{2^t-1}|x\rangle|f(x)\rangle. \tag{9}$$

Hence, we have a state which is a superposition of all the possible values of $f(x)$, stored in the second register qubits. After all these operations, we have to perform an inverse QFT. Now we are going to see how the periodicity of $f(x)$ is stored in the state and how we can extract it.

1. *Phase Estimation in Shor's algorithm*

We might notice that the circuit in Fig. 4 has the same structure as the phase estimation circuit in Fig. 2. The only difference is that now the state on the second register, $|1\rangle$, is not an eigenstate of the operator U. To see how we can link this with phase estimation, let us define the state

$$|u_s\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{\frac{-2\pi i s k}{r}}|a^k \pmod{N}\rangle. \tag{10}$$

This state is composed by all the possible values that $f(x) = a^x \pmod{N}$ can take, because it contains all the domain of $f(x)$ contained in one period (until $r-1$). Hence, U acting on this state only swaps one position of each one of the elements that compose the state. If we also take into account the phase of every state, then

$$U|u_s\rangle = e^{\frac{2\pi i s}{r}}|u_s\rangle. \tag{11}$$

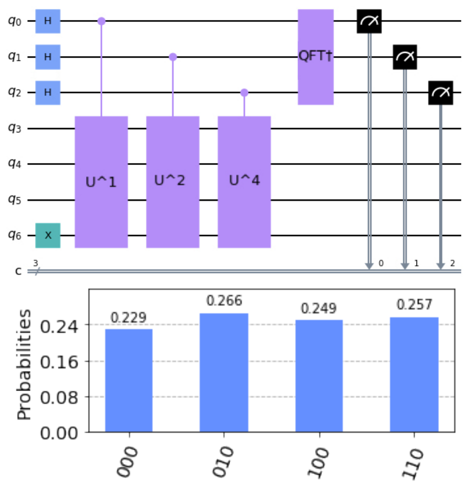

FIG. 6: Implementation of Shor's algorithm for 3 register qubits in an IBM simulator. Above, we show the circuit of the algorithm, where the U gate is the one in Fig. 5. Below, the results of the simulation with 3000 measurements on the first register qubits ($|q_2q_1q_0\rangle$).

So we have found another eigenstate of U with an interesting phase, which contains the periodicity of $f(x)$. If we knew how to prepare the eigenstate $|u_s\rangle$ for a certain s we could estimate s/r with the phase estimation algorithm. However, to do this we would need to know r, that is what we are looking for. Fortunately, this eigenstate fulfils

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = |1\rangle. \tag{12}$$

This is because the only term that survives is the one with $k=0$ in all the sums in Eq. (10). The other cancel each other due to the phases. Eq. (12) is powerful, because it says that, no matter what r we have, the state $|1\rangle$ will be a linear combination of eigenstates of U. Knowing this, we are able to interpret the circuit in Fig. 4.

First, we apply the Hadamard gates to the first register. Then, the U^{2^0} gate on the $|1\rangle$ state controlled by the first register qubit (q_0). We have the state

$$\frac{1}{\sqrt{2^t}}(|0\rangle^0 + |1\rangle^0 U_{|1\rangle})(|0\rangle^1+|1\rangle^1)...(|0\rangle^{t-1}+|1\rangle^{t-1})\otimes|1\rangle,$$

where the superscripts are referring to the qubits of the first register and $U_{|1\rangle}$ means that the operator only acts on the second register state. From now on, we are dropping the tensor product on the first register for simplicity. After applying the operator $U_{|1\rangle}$, we use Eq. (12) and the phase kickback mechanism discussed in Eq. (5). Then, we rewrite

$$\frac{1}{\sqrt{2^t r}}\sum_{s=0}^{r-1}\left[(|0\rangle^0 + e^{\frac{2\pi i s}{r}}|1\rangle^0)...(|0\rangle^{t-1}+|1\rangle^{t-1})\otimes|u_s\rangle\right].$$

If we take into account that $U^k|u_s\rangle = e^{\frac{2\pi i k s}{r}}|u_s\rangle$, and we continue applying all the controlled powers of the U gate, we obtain

$$\frac{1}{\sqrt{2^t r}}\sum_{s=0}^{r-1}\Big[\big(|0\rangle + e^{\frac{2\pi i 2^0 s}{r}}|1\rangle\big)\big(|0\rangle + e^{\frac{2\pi i 2^1 s}{r}}|1\rangle\big)...$$
$$\big(|0\rangle + e^{\frac{2\pi i 2^t s}{r}}|1\rangle\big)\Big]|u_s\rangle,$$

where we have defined $\phi_s = s/r$. This equation is very similar to (6), except for the sum and the order of the first register qubits, which is inverted due to the fact that now we are using qiskit notation, discussed previously. If we perform an inverse QFT, like in the phase estimation algorithm, we obtain a superposition of states with phases ϕ_s for $0 \le s \le r-1$,

$$\frac{1}{\sqrt{r}}\left(|2^t\cdot\frac{0}{r}\rangle + |2^t\cdot\frac{1}{r}\rangle + ... + |2^t\cdot\frac{r-1}{r}\rangle\right). \tag{13}$$

Therefore, if we measure the state on the first register qubits, we will obtain one of the equiprobable state $|2^t\cdot\frac{s}{r}\rangle$, for every s. Repeating the process and performing different measures on the previous state, we can use the continued fraction expansion classic algorithm to estimate the phase r.

IV. TESTING THE ALGORITHM WITH $N = 15$

We will use $t=3$ qubits on the first register, $[\log_2 15] = 4$ in the second and $a=7$. The circuit and the simulation results of Shor's Algorithm implementation are shown in Fig. 6.

The resulting states we obtain are $|0\rangle, |2\rangle, |4\rangle, |6\rangle$ (in decimal basis). Since we are obtaining $|2^t\phi_s\rangle$, we have to divide this result by 2^3. Therefore, the set of values ϕ_s

$$\left\{\frac{0}{2^3},\frac{2}{2^3},\frac{4}{2^3},\frac{6}{2^3}\right\} = \{0, 0.25, 0.5, 0.75\} = \left\{\frac{0}{1},\frac{1}{4},\frac{1}{2},\frac{3}{4}\right\},$$

let us infer that $r=4$, because it appears in two of the results. However, when $s=0$ and when s is not co-prime to r, we do not obtain directly s/r. For this reason it is important to measure as many times as necessary to make sure we will obtain all the possible output states.

In this case, r is not odd and $7^{4/2}+1 \neq$ (mod 15), then we can continue. We compute $\gcd(7^2 \pm 1, N) = \gcd(49\pm 1, 15)$=3 and 5. Therefore, we have solved the problem of factoring $N = 15$.

A. Factoring with different values of a

We have tested the circuit in Fig. 6 with a = 2, 7, 8, 9, 11 and 13. The results of the measurement executed in the simulator and on an IBM open-access QC (*'ibm-q-melboure16'*) are shown in Fig. 7.

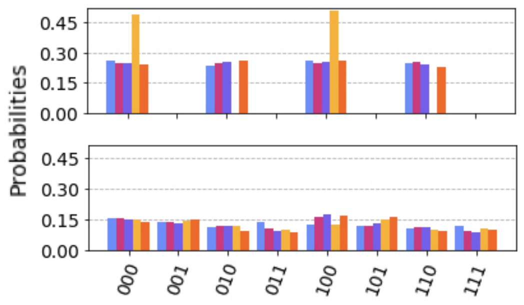

FIG. 7: Results obtained implementing Shor's algorithm using the circuit in Fig. 6 with different values of a. Above is shown the results of the simulation and below the results of IBM quantum computer *melbourne-16*. Blue corresponds to $a = 2$, magenta to $a = 7$, purple to $a = 8$, yellow to $a = 11$ and orange to $a = 13$.

The simulator gives us the probabilities we expected from the study of the algorithm. We obtain $r = 2$ for $a = 11$ and $r = 4$ for all the other cases. Nevertheless, the result measured in the QC has a significant probability in all of the eight possible states that can be measured. The worse case is for $a = 11$, in which we only had two possible resulting states in the simulator and we obtain almost the same probability in all the states in the QC measure. A similar behaviour is found for the rest of a values.

1. *Analysis of the results*

The results we have obtained are not good enough to factor a number. There might be some reasons that caused these outcomes. On the one hand, QC have to deal with noise, which can be caused by interference or lost of quantum coherence. This can severally affect the states and, therefore, the measurements. On the other hand, this algorithm requires a large amount of gates, i.e. a great number of U gates, which at the same time are composed by single-qubit gates, among others. This operation consumes a substantial amount of time and raises the probability of interference and decoherence. Even Shor in his article mentions that *"The bottleneck in the quantum factoring algorithm [...] is modular exponentiation."* [1].

2. *Number of qubits*

Shor's algorithm requires that we use t qubits in the first register, such that $N^2 < 2^t \leq 2N^2$. This condition ensures that we have at least N different values of x that gives the same value of $f(x)$, even for r approaching $N/2$. However, $N = 15$ is not a good example for the requisite of qubits. For $a = 7$, $a^{2^k} \pmod{15} = 1$ for $k \geq 2$. Therefore, only applying two controlled U gates, we will collect the periodicity of the function. Nevertheless, for larger numbers this will not be the case and we need more qubits to collect enough values of $f(x)$ [4].

Due to the actual limitation of open-access QC, we have performed the algorithm for only 3 register qubits.

V. CONCLUSIONS

In this work we have performed a detailed study of Shor's factoring algorithm. In the quantum implementation we have focused our attention on the role of phase estimation. Applying these concepts, we tried the algorithm with a classic example and then using a quantum computer to factor 15.

We have observed that classically we can perform the DFT of $f(x) = a^x \pmod{N}$ for obtaining the frequency and, therefore, the periodicity of the function. On the other hand, for the quantum implementation we have used a circuit that requires modular exponentiation and QFT. The results in the simulator are the expected. However, when we try the algorithm in an IBM QC, the results we obtain do not allow us to factor 15. We have argued that this can be caused due to the actual limitations of these systems. We hope near future advances in QC will allow the correct implementation of Shor's algorithm and thus profit its power.

Acknowledgments

I would like to thank Dr. Bruno Juliá Díaz for his guidance and support during the course of the project.

[1] P. Shor, *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, arXiv:quant-ph/9508027 (1996).
[2] M. A. Nielsen, and I. L. Chuang. *Quantum Computation and Quantum Information*, Cambridge University Press, (2010).
[3] *IBM Quantum Experience.* `https://quantum-computing.ibm.com`, (2016)
[4] D. Mermin. *Quantum Computer Science: An Introduction*, Cambridge University Press, (2007).
[5] Qiskit, *Shor's Algorithm*, `<https://qiskit.org/textbook/ch-algorithms/shor.html>`, (2021).
[6] Lieven M. *et al. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance*, Letters to nature, pages 883-887, (2001).
[7] L. Garrido's notes from Classical and Quantum Information Theory, UB Physics degree, (2020).