Shor's Algorithm for Factoring Large Integers*

C. Lavor†, L.R.U. Manssur‡, and R. Portugal‡

†Instituto de Matemática e Estatística
Universidade do Estado do Rio de Janeiro - UERJ
Rua São Francisco Xavier, 524, 6º andar, bl. D, sala 6018,
Rio de Janeiro, RJ, 20550-900, Brazil
e-mail: carlile@ime.uerj.br

‡Coordenação de Ciência da Computação
Laboratório Nacional de Computação Científica - LNCC
Av. Getúlio Vargas 333, Petrópolis, RJ, 25651-070, Brazil
e-mail: {leon,portugal}@lncc.br

February 1, 2008

Abstract

This work is a tutorial on Shor's factoring algorithm by means of a worked out example. Some basic concepts of Quantum Mechanics and quantum circuits are reviewed. It is intended for non-specialists which have basic knowledge on undergraduate Linear Algebra.

1 Introduction

In the last 30 years, the number of transistors per chip roughly doubled every 18 months, amounting to an exponentially growing power of classical computers. Eventually this statement (Moore's law) will be violated, since the transistor size will reach the limiting size of one atom in about 15 years. Even before that, disturbing quantum effects will appear.

Ordinarily one states that if an algorithm is inefficient, one simply waits for hardware efficient enough to run it. If the exponential increase in the power of classical computers becomes saturated, the class of inefficient algorithms will remain useless. From this pessimistic point of view, Computer Science seems to face very narrow limitations in the near future, coming from the physics underneath computer architecture.

It is important to keep in mind that a computer is a device governed by the laws of Physics. For decades, this fact was irrelevant. Computer Science emerged in a mathematical context and the specifics imposed by Physics were so few that most computer

*Contents based on lecture notes from graduate courses in Quantum Computation given at LNCC.

scientists paid no attention to them. One possible explanation for this state of matters is that computers work under the laws of classical physics, which are common sense.

In a seminal paper, Feynman [1] argued that the way classical computers work is a special case of some more general form allowed by the laws of Quantum Mechanics. He gave general arguments supporting the idea that a manifestly quantum device would be exponentially faster than a classical one. Subsequently, Deutsch [2] generalized the classical circuit model to its quantum counterpart and gave the first example of a quantum algorithm faster than its classical counterpart. Based on Deutsch's work, Simon [3] developed a quantum algorithm exponentially faster than its classical counterpart, taking advantage of entanglement, corroborating with Feynman's arguments.

The greatest success came with Shor's work [4]. He developed exponentially faster quantum algorithms for factoring integers and for finding discrete logarithms when compared to the known classical algorithms. Shor's algorithms allow one to render most current cryptographic methods useless, when a quantum computer of reasonable size is available.

This work is an introductory review of Shor's factoring algorithm. We have put all our efforts to write as clear as possible for non-specialists. We assume familiarity with undergraduate Linear Algebra, which is the main mathematical basis of Quantum Mechanics. Some previous knowledge of Quantum Mechanics is welcome, but not necessary for a persistent reader. The reader can find further material in [4, 5, 6, 7, 8].

Section 2 reviews basic notions of Quantum Mechanics necessary for Quantum Computation. Section 3 introduces the notion of quantum circuits and presents some basic examples. Section 4 describes how factorization can be reduced to order calculation and Section 5 gives a quantum algorithm for it. Section 6 shows the quantum Fourier transform. Section 7 gives an example and finally Section 8 shows the decomposition of the Fourier transform circuit in terms of the universal gates.

2 Review of Quantum Mechanics for Quantum Computation

In classical computers, a bit can assume only values 0 or 1. In quantum computers, the values 0 and 1 are replaced by the vectors $|0\rangle$ and $|1\rangle$. This notation for vectors is called the Dirac notation and is standard in Quantum Mechanics. The name bit is replaced by *qubit*, short of *quantum bit*. The difference between bits and qubits is that a qubit $|\psi\rangle$ can also be in a linear combination of the vectors $|0\rangle$ and $|1\rangle$,

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

where α and β are complex numbers. $|\psi\rangle$ is said to be a *superposition* of the vectors $|0\rangle$ and $|1\rangle$ with *amplitudes* α and β. Thus, $|\psi\rangle$ is a vector in a two-dimensional complex vector space, where $\{|0\rangle, |1\rangle\}$ forms an orthonormal basis, called the *computational basis* (see Fig. 1 in the real case). The state $|0\rangle$ is not the zero vector, but simply the first vector of the basis. The matrix representations of the vectors $|0\rangle$ and $|1\rangle$ are given by

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

In Quantum Mechanics, vectors are systematically called *states*. We use this term from now on.

The physical interpretation of $|\psi\rangle$ is that it coexists in two states: $|0\rangle$ and $|1\rangle$. It is similar to a coin that is partially heads up and partially tails up simultaneously. We cannot push further the analogy simply because quantum phenomena do not have a classical analogue in general. The state $|\psi\rangle$ can store a huge quantity of information in its coefficients α and β, but this information lives in the quantum level, which is microscopic (usually quantum effects appear in atomic dimensions). To bring quantum information to the classical level, one must measure the qubit. Quantum Mechanics tells us that the measurement process inevitably disturbs a qubit state, producing a non-deterministic collapse of $|\psi\rangle$ to either $|0\rangle$ or $|1\rangle$. One gets $|0\rangle$ with probability $|\alpha|^2$ or $|1\rangle$ with probability $|\beta|^2$. The non-deterministic collapse does not allow one to determine the values of α and β before the measurement. They are inaccessible via measurements unless one has many copies of the same state. Two successive measurements of the same qubit give the same output. If $|\alpha|^2$ and $|\beta|^2$ are probabilities and there are only two possible outputs, then

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2}$$

Calculating the norm of $|\psi\rangle$, Eq. (2) gives

$$|| \, |\psi\rangle \, || = \sqrt{|\alpha|^2 + |\beta|^2} = 1.$$

A measurement is not the only way that one can interact with a qubit. If one does not obtain any information about the state of the qubit, the interaction changes the values of α and β keeping the constraint (2). The most general transformation of this kind is a linear transformation U that takes unit vectors into unit vectors. Such a transformation is called *unitary* and can be defined by

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

where $U^\dagger = (U^*)^T$ ($*$ indicates complex conjugation and T indicates the transpose operation) and I is the 2×2 identity matrix.

So far we are dealing with one-qubit quantum computers. To consider the multiple qubit case, it is necessary to introduce the concept of *tensor product*. Suppose V and

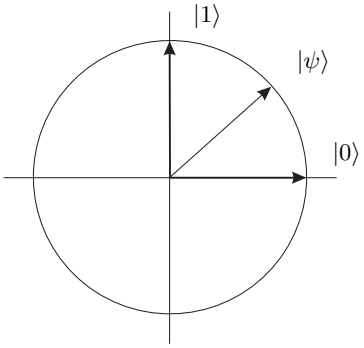

Figure 1: Computational basis for the case α, β real. In the general case (α, β complex) there is still a geometrical representation called the Bloch sphere (see [6] page 15).

W are complex vector spaces of dimensions m and n, respectively. The tensor product $V \otimes W$ is an mn-dimensional vector space. The elements of $V \otimes W$ are linear combinations of tensor products $|v\rangle \otimes |w\rangle$, satisfying the following properties ($z \in \mathbb{C}$, $|v\rangle, |v_1\rangle, |v_2\rangle \in V$, and $|w\rangle, |w_1\rangle, |w_2\rangle \in W$):

1. $z(|v\rangle \otimes |w\rangle) = (z|v\rangle) \otimes |w\rangle = |v\rangle \otimes (z|w\rangle)$,
2. $(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = (|v_1\rangle \otimes |w\rangle) + (|v_2\rangle \otimes |w\rangle)$,
3. $|v\rangle \otimes (|w_1\rangle + |w_2\rangle) = (|v\rangle \otimes |w_1\rangle) + (|v\rangle \otimes |w_2\rangle)$.

We use also the notations $|v\rangle|w\rangle$, $|v, w\rangle$ or $|vw\rangle$ for the tensor product $|v\rangle \otimes |w\rangle$. Note that the tensor product is non-commutative, so the notation must preserve the ordering.

Given two linear operators A and B defined on the vector spaces V and W, respectively, we can define the linear operator $A \otimes B$ on $V \otimes W$ as

$$(A \otimes B)(|v\rangle \otimes |w\rangle) = A|v\rangle \otimes B|w\rangle, \tag{3}$$

where $|v\rangle \in V$ and $|w\rangle \in W$. The matrix representation of $A \otimes B$ is given by

$$A \otimes B = \begin{bmatrix} A_{11}B & \cdots & A_{1m}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \cdots & A_{mm}B \end{bmatrix}, \tag{4}$$

where A is an $m \times m$ matrix and B is a $n \times n$ matrix (we are using the same notation for the operator and its matrix representation). So, the matrix $A \otimes B$ has dimension $mn \times mn$. For example, given

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

the tensor product $A \otimes B$ is

$$A \otimes B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

The formula (4) can also be used for non-square matrices, such as the tensor product of two vectors. For example, if we have a 2-qubit quantum computer and the first qubit is in the state $|0\rangle$ and the second is in the state $|1\rangle$, then the quantum computer is in the state $|0\rangle \otimes |1\rangle$, given by

$$|0\rangle \otimes |1\rangle = |01\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \tag{5}$$

The resulting vector is in a 4-dimensional vector space.

The general state $|\psi\rangle$ of a 2-qubit quantum computer is a superposition of the states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$,

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle, \tag{6}$$

with the constraint

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

Regarding the zeroes and ones as constituting the binary expansion of an integer, we can replace the representations of states

$$|00\rangle,\ |01\rangle,\ |10\rangle,\ |11\rangle,$$

by the shorter forms

$$|0\rangle,\ |1\rangle,\ |2\rangle,\ |3\rangle,$$

in decimal notation, which is handy in some formulas.

In general, the state $|\psi\rangle$ of an n-qubit quantum computer is a superposition of the 2^n states $|0\rangle$, $|1\rangle$, ..., $|2^n - 1\rangle$,

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

with amplitudes α_i constrained to

$$\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1.$$

Recall that the orthonormal basis $\{|0\rangle, \ldots, |2^n - 1\rangle\}$ is the computational basis in decimal notation. The state of an n-qubit quantum computer is a vector in a 2^n-dimensional complex vector space. When the number of qubits increases linearly, the dimension of the associated vector space increases exponentially. As before, a measurement of a generic state $|\psi\rangle$ yields the result $|i_0\rangle$ with probability $|\alpha_{i_0}|^2$, where $0 \leq i_0 < 2^n$. Usually, the measurement is performed qubit by qubit yielding zeroes or ones that are read together to form i_0. We stress again a very important feature of the measurement process. The state $|\psi\rangle$ as it is before measurement is inaccessible unless it is in the computational basis. The measurement process inevitably disturbs $|\psi\rangle$ forcing it to collapse to one vector of the computational basis. This collapse is non-deterministic, with the probabilities given by the squared norms of the corresponding amplitudes in $|\psi\rangle$.

If we have a 2-qubit quantum computer, the first qubit in the state

$$|\varphi\rangle = a|0\rangle + b|1\rangle$$

and the second in the state

$$|\psi\rangle = c|0\rangle + d|1\rangle,$$

then the state of the quantum computer is the tensor product

$$\begin{aligned} |\varphi\rangle \otimes |\psi\rangle &= (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) \\ &= ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle. \end{aligned} \tag{7}$$

Note that a general 2-qubit state (6) is of the form (7) if and only if

$$\begin{aligned} \alpha &= ac, \\ \beta &= ad, \\ \gamma &= bc, \\ \delta &= bd. \end{aligned}$$

From these equalities we have that a general 2-qubit state (6) is of the form (7) if and only if

$$\alpha\delta = \beta\gamma.$$

Thus, the general 2-qubit state is not necessarily a product of two one-qubit states. Such non-product states of two or more qubits are called *entangled* states, for example, $(|00\rangle + |11\rangle)/\sqrt{2}$. The entangled states play an essential role in quantum computation. Quantum computers that do not use entanglement cannot be exponentially faster than classical computers. On the other hand, a naive use of entanglement does not guarantee any improvements.

A complex vector space V is a Hilbert space if there is an *inner product*, written in the form $\langle\varphi|\psi\rangle$, defined by the following rules ($a, b \in \mathbb{C}$ and $|\varphi\rangle, |\psi\rangle, |u\rangle, |v\rangle \in V$):

1. $\langle\psi|\varphi\rangle = \langle\varphi|\psi\rangle^*$,
2. $\langle\varphi|(a|u\rangle + b|v\rangle)\rangle = a\langle\varphi|u\rangle + b\langle\varphi|v\rangle$,
3. $\langle\varphi|\varphi\rangle > 0$ if $|\varphi\rangle \neq 0$.

The *norm* of a vector $|\varphi\rangle$ is given by

$$|| \, |\varphi\rangle \, || = \sqrt{\langle\varphi|\varphi\rangle}.$$

The notation $\langle\varphi|$ is used for the *dual vector* to the vector $|\varphi\rangle$. The dual is a linear operator from the vector space V to the complex numbers, defined by

$$\langle\varphi|(|v\rangle) = \langle\varphi|v\rangle, \quad \forall |v\rangle \in V.$$

Given two vectors $|\varphi\rangle$ and $|\psi\rangle$ in a vector space V, there is also an *outer product* $|\psi\rangle\langle\varphi|$, defined as a linear operator on V satisfying

$$(|\psi\rangle\langle\varphi|)|v\rangle = |\psi\rangle\langle\varphi|v\rangle, \quad \forall |v\rangle \in V.$$

If $|\varphi\rangle = a|0\rangle + b|1\rangle$ and $|\psi\rangle = c|0\rangle + d|1\rangle$, then the matrix representations for inner and outer products are:

$$\begin{aligned} \langle\varphi|\psi\rangle &= \begin{bmatrix} a^* & b^* \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = a^*c + b^*d, \\ |\varphi\rangle\langle\psi| &= \begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} c^* & d^* \end{bmatrix} = \begin{bmatrix} ac^* & ad^* \\ bc^* & bd^* \end{bmatrix}. \end{aligned}$$

The matrix of the outer product is obtained by usual matrix multiplication of a column matrix by a row matrix. But in this case, we can replace the matrix multiplication by

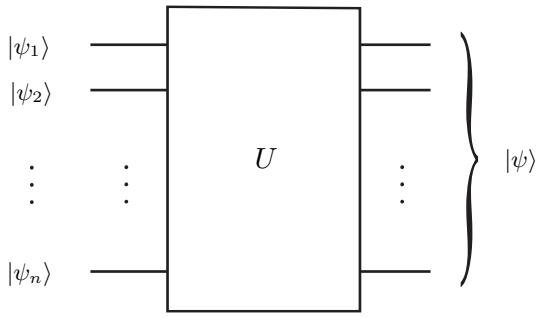

Figure 2: The sketch of the quantum computer. We consider the input nonentangled, which is reasonable in general. On the other hand, the output is entangled in general. The measurement of the state $|\psi\rangle$, not shown here, returns zeroes and ones.

the tensor product, i.e., $|\varphi\rangle\langle\psi| = |\varphi\rangle \otimes \langle\psi|$ (notice the complex conjugation in the process of taking the dual).

After the above review, we are ready to outline the quantum computer. In Fig. 2, we are taking a nonentangled input, what is quite reasonable. In fact, $|\psi_i\rangle$ is either $|0\rangle$ or $|1\rangle$ generally. $|\psi\rangle$, on the right hand side of Fig. 2, is the result of the application of a unitary operator U on the input. The last step is the measurement of the states of each qubit, which returns zeroes and ones that form the final result of the quantum calculation. Note that there is, in principle, an infinite number of possible operators U, which are unitary $2^n \times 2^n$ matrices.

3 Quantum Circuits

Let us start with one-qubit gates. In the classical case there is only one possibility, which is the NOT gate. The NOT gate inverts the bit value: 0 goes to 1 and vice-versa. The straightforward generalization to the quantum case is given in Fig. 3, where X is the unitary operator

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

So, if the input $|\psi\rangle$ is $|0\rangle$, the output is $|1\rangle$ and vice-versa. But now we can have a situation with no classical counterpart. The state $|\psi\rangle$ can be a superposition of states $|0\rangle$ and $|1\rangle$. The general case is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and the corresponding output is $\alpha|1\rangle + \beta|0\rangle$.

The gate X is not the only one-qubit gate. There are infinitely many, since there are an infinite number of 2×2 unitary matrices. In principle, any unitary operation can be

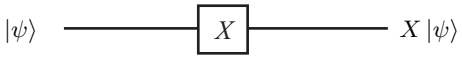

Figure 3: Quantum NOT gate.

implemented in practice. The *Hadamard* gate is another important one-qubit gate, given by

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

It is easy to see that

$$\begin{aligned} H|0\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \\ H|1\rangle &= \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \end{aligned}$$

If the input is $|0\rangle$, the Hadamard gate creates a superposition of states with equal weights. This is a general feature, valid for two or more qubits. Let us analyze the 2-qubit case.

The first example of a 2-qubit gate is $H \otimes H$:

$$\begin{aligned} H^{\otimes 2}|0\rangle|0\rangle &= (H \otimes H)(|0\rangle \otimes |0\rangle) = H|0\rangle \otimes H|0\rangle \\ &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \\ &= \frac{1}{2}(|0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle + |1\rangle|1\rangle) \\ &= \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle). \end{aligned}$$

The result is a superposition of all basis states with equal weights. More generally, the Hadamard operator applied to the n-qubit state $|0\rangle$ is

$$\begin{aligned} H^{\otimes n}|0\rangle &= H^{\otimes n}\,|0, ..., 0\rangle = (H\,|0\rangle)^{\otimes n} \\ &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes n} \\ &= \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle\,. \end{aligned} \tag{8}$$

Thus, the tensor product of n Hadamard operators produces an equally weighted superposition of all computational basis states, when the input is the state $|0\rangle$. This state is useful for applying quantum parallelism, as we will see ahead.

Another important 2-qubit quantum gate is the CNOT gate. It has two input qubits, the control and the target qubit, respectively. The target qubit is flipped only if the control qubit is set to 1, that is,

$$\begin{aligned} |00\rangle &\rightarrow |00\rangle, \\ |01\rangle &\rightarrow |01\rangle, \\ |10\rangle &\rightarrow |11\rangle, \\ |11\rangle &\rightarrow |10\rangle. \end{aligned} \tag{9}$$

The action of the CNOT gate can also be represented by

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

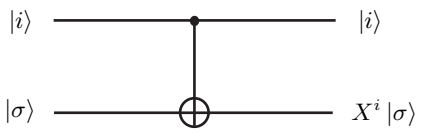

Figure 4: CNOT gate. $|i\rangle$ can be either $|0\rangle$ or $|1\rangle$. The general case is obtained by linearity.

where $\oplus$ is addition modulo 2. Now, let us obtain its matrix representation. Performing the same calculations that yield Eq. (5), we have

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

Thus, from (9) and (10), the matrix representation U_{CNOT} of the CNOT gate is

$$U_{\text{CNOT}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Fig. 4 describes the CNOT gate, where $|i\rangle$ is either $|0\rangle$ or $|1\rangle$. The figure could lead one to think that the output is always nonentangled, but that is not true, since if the first qubit is in a more general state given by $a\,|0\rangle + b\,|1\rangle$, then the output will be $a\,|0\rangle\,|\sigma\rangle + b\,|1\rangle\,X\,|\sigma\rangle$, which is entangled in general. The input can be entangled as well.

We have seen two examples of 2-qubit gates. The general case is a 4×4 unitary matrix. Gates that are the direct product of other gates, such as $H \otimes H$, do not produce entanglement. If the input is nonentangled, the output is not too. On the other hand, the output of the CNOT gate can be entangled while the input is nonentangled.

CNOT and one-qubit gates form a universal set of gates. This means that any other gate, operating on 2 or more qubits, can be written as compositions and direct products of CNOT and one-qubit gates [6].

At the end of Section 2, we gave a general outline of the quantum computer (Fig. 2) based on the action of a unitary operator U. In the present section, we have seen that in general U can be broken up in terms of smaller gates. This decomposition is useful because it corresponds to the natural steps that describe an algorithm. So, a quantum algorithm consists of a sequence of unitary operators acting on sets of qubits. These unitary operators multiplied together form the operator U of Fig. 2.

More details on quantum circuits can be found in [6, 9].

4 Factorization can be reduced to order calculation

Let us describe Shor's algorithm for finding the prime factors of a composite number N. Think of a large number such as one with 300 digits in decimal notation, since such

numbers are used in cryptography. Though N is large, the number of qubits necessary to store it is small. In general $\log_2 N$ is not an integer, so let us define

$$n = \lceil \log_2 N \rceil .$$

A quantum computer with n qubits can store N or any other positive integer less than N. With a little thought, we see that the number of prime factors of N is at most n. If both the number of qubits and the number of factors are less than or equal to n, then it is natural to ask if there is an algorithm that factors N in a number of steps which is polynomial in n. More technically, the question is: is there a factorization algorithm in the complexity class $\mathcal{P}$ [10]?

Reduction of factorization of N to the problem of finding the *order* of an integer x less than N is as follows. If x and N have common factors, then $\mathrm{GCD}(x, N)$ gives a factor of N, therefore it suffices to investigate the case when x is coprime to N. The order of x modulo N is defined as the least positive integer r such that

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

If r is even, we can define y by

$$x^{r/2} \equiv y \mod N.$$

The above notation means that y is the remainder of $x^{r/2}$ divided by N and, by definition, $0 \leq y < N$. Note that y satisfies $y^2 \equiv 1$ modulo N, or equivalently $(y-1)(y+1) \equiv 0$ modulo N, which means that N divides $(y-1)(y+1)$. If $1 < y < N-1$, the factors $y-1$ and $y+1$ satisfy $0 < y-1 < y+1 < N$, therefore N cannot divide $y-1$ nor $y+1$ separately. The only alternative is that both $y-1$ and $y+1$ have factors of N (that yield N by multiplication). So, $\mathrm{GCD}(y-1, N)$ and $\mathrm{GCD}(y+1, N)$ yield non trivial factors of N (GCD stands for the greatest common divisor). If N has remaining factors, they can be calculated applying the algorithm recursively.

Consider $N = 21$ as an example. The sequence of equivalences

$$\begin{aligned} 2^4 &\equiv 16 \mod 21 \\ 2^5 &\equiv 11 \mod 21 \\ 2^6 &\equiv 11 \times 2 \equiv 1 \mod 21 \end{aligned}$$

show that the order of 2 modulo 21 is $r = 6$. Therefore, $y \equiv 2^3 \equiv 8$ modulo 21. $y-1$ yields the factor 7 and $y+1$ yields the factor 3 of 21.

In summary, if we pick up at random a positive integer x less than N and calculate $\mathrm{GCD}(x, N)$, either we have a factor of N or we learn that x is coprime to N. In the latter case, if x satisfies the conditions (1) its order r is even, and (2) $0 < y-1 < y+1 < N$, then $\mathrm{GCD}(y-1, N)$ and $\mathrm{GCD}(y+1, N)$ yield factors of N. If one of the conditions is not true, we start over until finding a proper candidate x. The method would not be useful if these assumptions were too restrictive, but fortunately that is not the case. The method sistematically fails if N is a power of some odd prime, but an alternative efficient classical algorithm for this case is known. If N is even, we can keep dividing by 2 until the result turns out to be odd. It remains to apply the method for odd composite integers that are not a power of some prime number. It is cumbersome to prove that the probability of finding x coprime to N satisfying the conditions (1) and (2) is high; in fact

this probability is $1 - 1/2^{k-1}$, where k is the number of prime factors of N. In the worst case (N has 2 factors), the probability is greater than or equal to 1/2 (see the proof in Appendix B of [5]).

At first sight, it seems that we have just described an efficient algorithm to find a factor of N. That is not true, since it is not known an efficient classical algorithm to calculate the order of an integer x modulo N. On the other hand, there is (after Shor's work) an efficient quantum algorithm. Let us describe it.

5 Quantum algorithm to calculate the order

Consider the circuit of Fig. 5. It calculates the order r of the positive integer x less than N, coprime to N. V_x is the unitary linear operator

$$V_x(|j\rangle\,|k\rangle) = |j\rangle\,\left|k + x^j\right\rangle, \tag{11}$$

where $|j\rangle$ and $|k\rangle$ are the states of the first and second registers, respectively. The arithmetical operations are performed modulo N, so $0 \leq k + x^j < N$. DFT is the Discrete Fourier Transform operator which will be described ahead.

The first register has t qubits, where t is generally chosen such that $N^2 \leq 2^t < 2N^2$, for reasons that will become clear later on [4]. As an exception, if the order r is a power of 2, then it is enough to take $t = n$. In this section we consider this very special case and leave the general case for Section 7. We will keep the variable t in order to generalize the discussion later on.

The states of the quantum computer are indicated by $|\psi_0\rangle$ to $|\psi_5\rangle$ in Fig. 5. The initial state is

$$|\psi_0\rangle = \underbrace{|0\ldots0\rangle}_{t}\,\underbrace{|0\ldots0\rangle}_{n}.$$

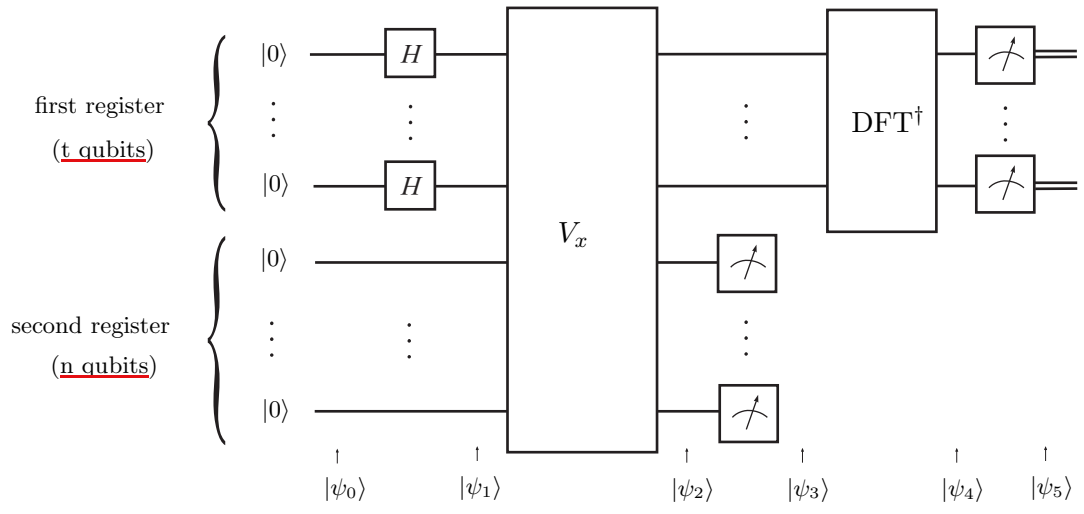

Figure 5: Quantum circuit for finding the order of the positive integer x modulo N.

The application of the Hadamard operator

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

on each qubit of the first register yields (see Eq. (8))

$$|\psi_1\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle\,|0\rangle\,. \tag{12}$$

The first register is then in a superposition of all states of the computational basis with equal amplitudes given by $\frac{1}{\sqrt{2^t}}$. Now we call the reader's attention to what happens when we apply V_x to $|\psi_1\rangle$:

$$\begin{aligned} |\psi_2\rangle &= V_x\,|\psi_1\rangle \\ &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} V_x(|j\rangle\,|0\rangle) \\ &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle\,\left|x^j\right\rangle\,. \end{aligned} \tag{13}$$

The state $|\psi_2\rangle$ is a remarkable one. Because V_x is linear, it acts on all $|j\rangle\,|0\rangle$ for 2^t values of j, so this generates all powers of x simultaneously. This feature is called *quantum parallelism*. Some of these powers are 1, which correspond to the states $|0\rangle\,|1\rangle$, $|r\rangle\,|1\rangle$, $|2r\rangle\,|1\rangle$, ..., $\left|\left(\frac{2^t}{r}-1\right)r\right\rangle|1\rangle$. This explains the choice (11) for V_x. Classically, one would calculate successively x^j, for j starting from 2 until reaching $j = r$. *Quantumly*, one can calculate all powers of x with just one application of V_x. At the quantum level, the values of j that yield $x^j \equiv 1$ modulo N are "known". But this quantum information is not fully available at the classical level. A classical information of a quantum state is obtained by practical measurements and, at this point, it does not help if we measure the first register, since all states in the superposition (13) have equal amplitudes. The first part of the strategy to find r is to observe that the first register of the states $|0\rangle\,|1\rangle$, $|r\rangle\,|1\rangle$, $|2r\rangle\,|1\rangle$, ..., $\left|2^t - r\right\rangle|1\rangle$ is periodic. So the information we want is a period. In order to simplify the calculation, let us measure the second register. Before doing this, we will rewrite $|\psi_2\rangle$ collecting equal terms in the second register. Since x^j is a periodic function with period r, substitute $ar + b$ for j in Eq. (13), where $0 \le a \le (2^t/r) - 1$ and $0 \le b \le r - 1$. Recall that we are supposing that $t = n$ and r is a power of 2, therefore r divides 2^t. Eq. (13) is converted to

$$|\psi_2\rangle = \frac{1}{\sqrt{2^t}} \sum_{b=0}^{r-1} \left(\sum_{a=0}^{\frac{2^t}{r}-1} |ar+b\rangle \right) \left|x^b\right\rangle\,. \tag{14}$$

In the second register, we have substituted x^b for x^{ar+b}, since $x^r \equiv 1$ modulo N. Now the second register is measured. Any output x^0, x^1, ..., x^{r-1} can be obtained with equal

probability. Suppose that the result is x^{b_0}. The state of the quantum computer is now

$$|\psi_3\rangle = \sqrt{\frac{r}{2^t}} \left(\sum_{a=0}^{\frac{2^t}{r}-1} |ar + b_0\rangle \right) \left| x^{b_0} \right\rangle . \tag{15}$$

Note that after the measurement, the constant is renormalized to $\sqrt{r/2^t}$, since there are $2^t/r$ terms in the sum (15). Fig. 6 shows the probability of obtaining the states of the computational basis upon measuring the first register. The probabilities form a periodic function with period r. Their values are zero except for the states $|b_0\rangle$, $|r + b_0\rangle$, $|2r + b_0\rangle$, ..., $\left|2^t - r + b_0\right\rangle$.

How can one find out the period of a function efficiently? The answer is in the Fourier transform. The Fourier transform of a periodic function with period r is a new periodic function with period proportional to $1/r$. This makes a difference for finding r. The Fourier transform is the second and last part of the strategy. The whole method relies on an efficient quantum algorithm for calculating the Fourier transform, which is not available classically. In Section 8, we show that the Fourier transform is calculated efficiently in a quantum computer.

6 The quantum discrete Fourier transform

The Fourier transform of the function $F : \{0, \ldots, N-1\} \to \mathbb{C}$ is a new function $\tilde{F} : \{0, \ldots, N-1\} \to \mathbb{C}$ defined as

$$\tilde{F}(k) = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i j k/N} F(j). \tag{16}$$

We can apply the Fourier transform either on a function or on the states of the computational basis. The Fourier transform applied to the state $|k\rangle$ of the computational basis

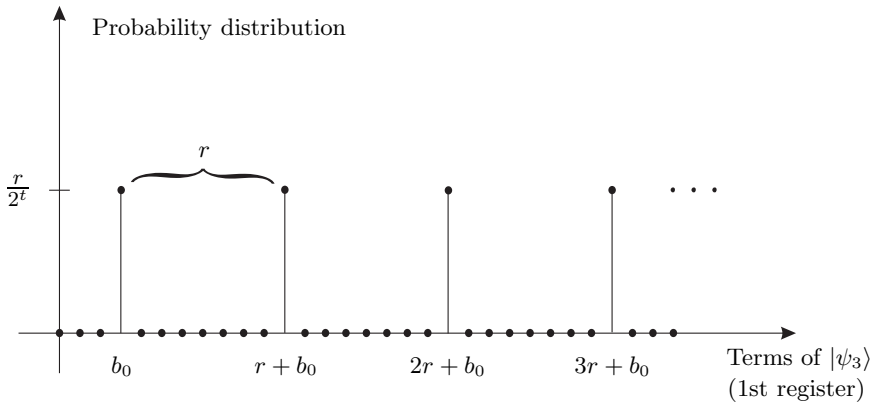

Figure 6: Probability distribution of $|\psi_3\rangle$ measured in the computational basis (for the case $b_0 = 3$ and $r = 8$). The horizontal axis has 2^t points. The number of peaks is $2^t/r$ and the period is r.

$\{|0\rangle, \ldots, |N-1\rangle\}$ is

$$\mathrm{DFT}(|k\rangle) = |\psi_k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi ijk/N} |j\rangle , \tag{17}$$

where the set $\{|\psi_k\rangle : k = 0, \ldots, N-1\}$ forms a new orthonormal basis. The Fourier transform is a unitary linear operator. So, if we know how it acts on the states of the computational basis, we also know how it acts on a generic state

$$|\psi\rangle = \sum_{a=0}^{N-1} F(a) |a\rangle .$$

The Fourier transform of $|\psi\rangle$ can be performed indistinctly using either (16) or (17). We will use the latter.

To prove that $\{|\psi_k\rangle : k = 0, \ldots, N-1\}$ is an orthonormal basis, i.e.,

$$\langle \psi_{k'} | \psi_k \rangle = \delta_{k'k},$$

we can use the identity

$$\frac{1}{N} \sum_{j=0}^{N-1} e^{2\pi ijk/N} = \begin{cases} 1 \text{ if } k \text{ is a multiple of } N \\ 0 \text{ otherwise,} \end{cases} \tag{18}$$

which is useful in the context of Fourier transforms. It is easy to verify that (18) is true. If k is a multiple of N, then $e^{2\pi ijk/N} = 1$ and the first case of the identity follows. If k is not a multiple of N, (18) is true even if N is not a power of 2. Fig. 7 shows each term $e^{2\pi ijk/N}$ $(j = 0, ..., 6)$ for the case $k = 1$ and $N = 7$, as vectors in the complex plane. Note that the sum of vectors must be zero by a symmetry argument: the distribution of vectors is isotropic. Usually it is said that the interference is destructive in this case. Using this identity, we can define the inverse Fourier transform, which is similar to (17), just with a minus sign on the exponent. Note that $\mathrm{DFT}^{-1} = \mathrm{DFT}^{\dagger}$, since DFT is a unitary operator.

We will present the details of a quantum circuit to perform the Fourier transform in Section 8. Now we will continue the calculation process of the circuit of Fig. 5. We are ready to find out the next state of the quantum computer—$|\psi_4\rangle$. Applying the inverse Fourier transform on the first register, using Eq. (17) and the linearity of $\mathrm{DFT}^{\dagger}$, we obtain

$$\begin{aligned} |\psi_4\rangle &= \mathrm{DFT}^{\dagger}(|\psi_3\rangle) \\ &= \sqrt{\frac{r}{2^t}} \sum_{a=0}^{\frac{2^t}{r}-1} \left(\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{-2\pi ij(ar+b_0)/2^t} |j\rangle \right) \left| x^{b_0} \right\rangle . \end{aligned}$$

Inverting the summation order, we have

$$|\psi_4\rangle = \frac{1}{\sqrt{r}} \left(\sum_{j=0}^{2^t-1} \left[\frac{1}{2^t/r} \sum_{a=0}^{\frac{2^t}{r}-1} e^{\frac{-2\pi ija}{2^t/r}} \right] e^{-2\pi ijb_0/2^t} |j\rangle \right) \left| x^{b_0} \right\rangle . \tag{19}$$

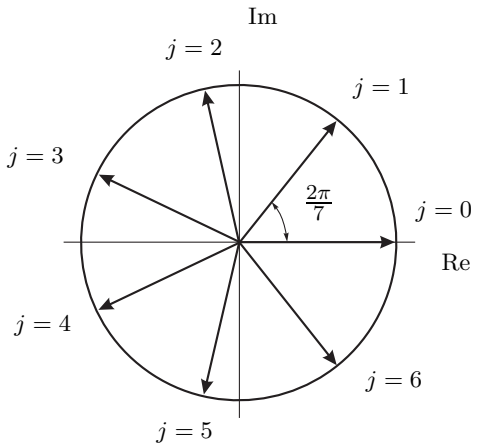

Figure 7: Vectors $e^{2\pi ij/7}$ $(j = 0, ..., 6)$ in the complex plane. Their sum is zero by symmetry arguments. This is an example of Eq. (18) for $N = 7$, $k = 1$.

Using (18), we see that the expression in square brackets is not zero if and only if $j = k2^t/r$, with $k = 0, ..., r-1$. When j takes such values, the expression in the square brackets is equal to 1. So we have

$$|\psi_4\rangle = \frac{1}{\sqrt{r}} \left(\sum_{k=0}^{r-1} e^{-2\pi i \frac{k}{r} b_0} \left| \frac{k2^t}{r} \right\rangle \right) \left| x^{b_0} \right\rangle . \tag{20}$$

In order to find r, the expression for $|\psi_4\rangle$ has two advantages over the expression for $|\psi_3\rangle$ (Eq. (15)): r is in the denominator of the ket label and the random parameter b_0 moved from the ket label to the exponent occupying now a harmless place. Fig. 8 shows the probability distribution of $|\psi_4\rangle$ measured in the computational basis. Measuring the first register, we get the value $k_0 2^t/r$, where k_0 can be any number between 0 and $r-1$ with equal probability (the peaks in Fig. 8). If we obtain $k_0 = 0$, we have no clue at

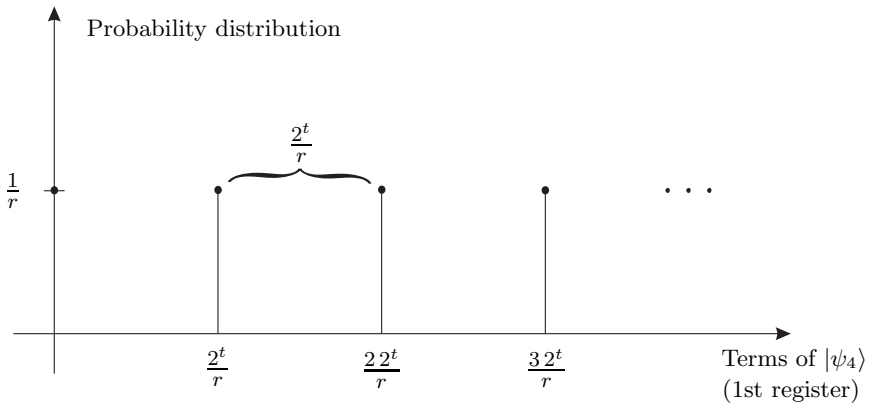

Figure 8: Probability distribution of $|\psi_4\rangle$ measured in the computational basis. The horizontal axis has 2^t points, only the non-null terms are shown. The number of peaks is r and the period is $2^t/r$.

all about r, and the algorithm must be run again. If $k_0 \neq 0$, we divide $k_0 2^t/r$ by 2^t, obtaining k_0/r. Neither k_0 nor r are known. If k_0 is coprime to r, we simply select the denominator.

If k_0 and r have a common factor, the denominator of the reduced fraction k_0/r is a factor of r but not r itself. Suppose that the denominator is r_1. Let $r = r_1 r_2$. Now the goal is to find r_2, which is the order of x^{r_1}. We run again the quantum part of the algorithm to find the order of x^{r_1}. If we find r_2 in the first round, the algorithm halts, otherwise we apply it recursively. The recursive process does not last, because the number of iterations is less than or equal to $\log_2 r$.

Take $N = 15$ as an example, which is the least nontrivial composite number. The set of numbers less than 15, coprime to 15 is $\{1, 2, 4, 7, 8, 11, 13, 14\}$. The numbers in the set $\{4, 11, 14\}$ have order 2 and the numbers in the set $\{2, 7, 8, 13\}$ have order 4. Therefore, in any case r is a power of 2 and the factors of $N = 15$ can be found in a 8-bit quantum computer ($t+n = 2\lceil \log_2 15 \rceil = 8$). The authors of [11] used a 7-qubit quantum computer, bypassing part of the algorithm.

7 Generalization by means of an example

In the previous sections, we have considered a special case when the order r is a power of 2 and $t = n$ (t is the number of qubits in the first register—see Fig. 5—and $n = \lceil \log_2 N \rceil$). In this section, we consider the factorization of $N = 21$, that is the next nontrivial composite number. We must choose t such that 2^t is between N^2 and $2N^2$, which is always possible [4]. For $N = 21$, the smallest value of t is 9. This is the simplest example allowed by the constraints, but enough to display all properties of Shor's algorithm.

The first step is to pick up x at random such that $1 < x < N$, and to test whether x is coprime to N. If not, we easily find a factor of N by calculating $\mathrm{GCD}(x, N)$. If yes, the quantum part of the algorithm starts. Suppose that $x = 2$ has been chosen. The goal is to find out that the order of x is $r = 6$. The quantum computer is initialized in the state

$$|\psi_0\rangle = |0\rangle\,|0\rangle\,,$$

where the first register has $t = 9$ qubits and the second has $n = 5$ qubits. Next step is the application of $H^{\otimes 9}$ on the first register yielding (see Eq. (12))

$$|\psi_1\rangle = \frac{1}{\sqrt{512}} \sum_{j=0}^{511} |j\rangle\,|0\rangle\,.$$

The next step is the application of V_x (defined in (11)), which yields

$$\begin{aligned} |\psi_2\rangle &= \frac{1}{\sqrt{512}} \sum_{j=0}^{511} |j\rangle\,\left|2^j \bmod N\right\rangle \\ &= \frac{1}{\sqrt{512}} \Big(|0\rangle\,|1\rangle + |1\rangle\,|2\rangle + |2\rangle\,|4\rangle + |3\rangle\,|8\rangle + |4\rangle\,|16\rangle + |5\rangle\,|11\rangle + \\ &\qquad |6\rangle\,|1\rangle + |7\rangle\,|2\rangle + |8\rangle\,|4\rangle + |9\rangle\,|8\rangle + |10\rangle\,|16\rangle + |11\rangle\,|11\rangle + \\ &\qquad |12\rangle\,|1\rangle + \ldots \Big). \end{aligned}$$

Notice that the above expression has the following pattern: the states of the second register of each "column" are the same. Therefore we can rearrange the terms in order to collect the second register:

$$\begin{aligned}|\psi_2\rangle \;=\; \frac{1}{\sqrt{512}}\Big[&\big(\,|0\rangle+\;|6\rangle+|12\rangle+\ldots+|504\rangle+|510\rangle\,\big)\,|1\rangle+\\ &\big(\,|1\rangle+\;|7\rangle+|13\rangle+\ldots+|505\rangle+|511\rangle\,\big)\,|2\rangle+\\ &\big(\,|2\rangle+\;|8\rangle+|14\rangle+\ldots+|506\rangle\,\big)\;\;|4\rangle+\\ &\big(\,|3\rangle+\;|9\rangle+|15\rangle+\ldots+|507\rangle\,\big)\;\;|8\rangle+\\ &\big(\,|4\rangle+|10\rangle+|16\rangle+\ldots+|508\rangle\,\big)\,|16\rangle+\\ &\big(\,|5\rangle+\;|11\rangle+|17\rangle+\ldots+|509\rangle\,\big)\,|11\rangle\Big].\end{aligned} \tag{21}$$

This feature was made explicit in Eq. (14). Because the order is not a power of 2, here there is a small difference: the first two lines of Eq. (21) have 86 terms, while the remaining ones have 85.

Now one measures the second register[1], yielding one of the following numbers equiprobably: $\{1, 2, 4, 8, 16, 11\}$. Suppose that the result of the measurement is 2, then

$$|\psi_3\rangle = \frac{1}{\sqrt{86}}\left(|1\rangle+|7\rangle+|13\rangle+\ldots+|505\rangle+|511\rangle\right)|2\rangle\,. \tag{22}$$

Notice that the state $|\psi_3\rangle$ was renormalized in order to have unit norm. It does not matter what is the result of the measurement; what matters is the periodic pattern of (22). The period of the states of the first register is the solution to the problem and the Fourier transform can reveal the value of the period. So, the next step is the application of the inverse Fourier transform on the first register of $|\psi_3\rangle$:

$$\begin{aligned}|\psi_4\rangle \;&=\; \mathrm{DFT}^\dagger(|\psi_3\rangle)\\ &=\; \mathrm{DFT}^\dagger\left(\frac{1}{\sqrt{86}}\sum_{a=0}^{85}|6a+1\rangle\right)|2\rangle\\ &=\; \frac{1}{\sqrt{512}}\sum_{j=0}^{511}\left(\left[\frac{1}{\sqrt{86}}\sum_{a=0}^{85}e^{-2\pi i\frac{6ja}{512}}\right]e^{-2\pi i\frac{j}{512}}\,|j\rangle\right)|2\rangle\,,\end{aligned} \tag{23}$$

where we have used Eq. (17) and have rearranged the sums. The last equation is similar to Eq. (19), but with an important difference. In Section 5, we were assuming that r divides 2^t. This is not true in the present example (6 does not divide 512), therefore we cannot use the identity (18) to simplify the term in brackets in Eq. (23). This term never vanishes, but its main contribution is still around $j = 0,\ 85,\ 171,\ 256,\ 341,\ 427$, which are obtained rounding $512k_0/6$ for k_0 from 0 to 5—compare to the discussion that follows Eq. (20). To see this, let us plot the probability of getting the result j (in the interval 0 to 511) by measuring the first register of the state $|\psi_4\rangle$. From (23), we have

[1] As measurements can always be performed in the end (see [6] page 186), this step is not necessary. It is commonly used to simplify the expressions that follow.

that the probability is

$$\text{Prob}(j) = \frac{1}{512 \times 86} \left| \sum_{a=0}^{85} e^{-2\pi i \frac{6ja}{512}} \right|^2 . \tag{24}$$

The plot of Prob(j) is shown in Fig. 9. We see the peaks around $j = 0$, 85, 171, 256, 341, 427, indicating a high probability of getting one of these values, or some value very close to them. In between, the probability is almost zero. The sharpness of the peaks depends on t (number of qubits in the first register). The lower limit $2^t \geq N^2$ ensures a high probability in measuring a value of j carrying the desired information. A careful analysis of the expression (24) is performed in [12] and a meticulous study of the peak form is performed in [13].

Let us analyze the possible measurement results. If we get $j = 0$ (first peak), the algorithm has failed in this round. It must be run again. We keep $x = 2$ and rerun the quantum part of the algorithm. The probability of getting $j = 0$ is low: from Eq. (24) we have that Prob(0) $= 86/512 \approx 0.167$. Now suppose we get $j = 85$ (or any value in the second peak). We divide by 512 yielding 85/512, which is a rational approximation of $k_0/6$, for $k_0 = 1$. How can we obtain r from 85/512?

The method of continued fraction approximation allows one to extract the desired information. A general continued fraction expansion of a rational number j_1/j_2 has the form

$$\frac{j_1}{j_2} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{\ldots + \cfrac{1}{a_p}}},$$

usually represented as $[a_0, a_1, ..., a_p]$, where a_0 is a non-negative integer and $a_1, ..., a_p$ are positive integers. The q-th convergent ($0 \leq q \leq p$) is defined as the rational number $[a_0, a_1, ..., a_q]$. It is an approximation to j_1/j_2 and has a denominator smaller than j_2.

This method is easily applied by inversion of the fraction followed by integer division with rational remainder. Inverting 85/512 yields 512/85, which is equal to $6 + 2/85$. We

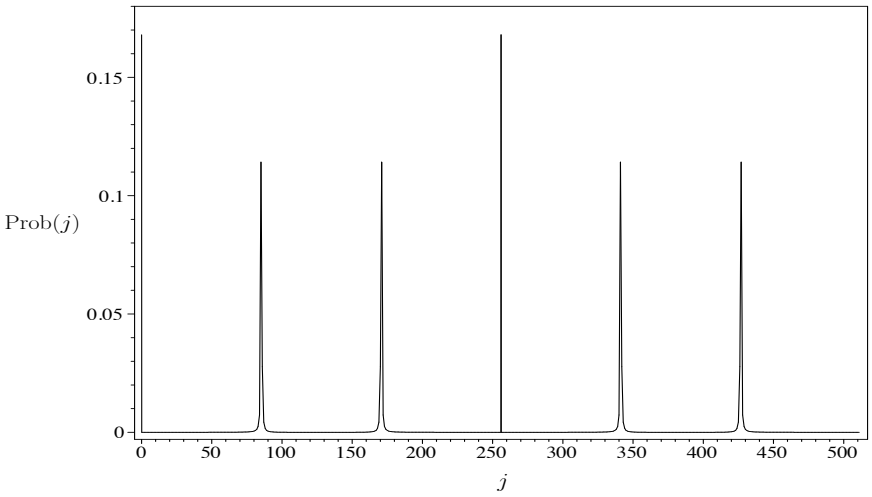

Figure 9: Plot of Prob(j) against j. Compare to the plot of Fig. 8, where peaks are not spread and have the same height.

repeat the process with 2/85 until we get numerator 1. The result is

$$\frac{85}{512} = \frac{1}{6 + \frac{1}{42+\frac{1}{2}}}.$$

So, the convergents of 85/512 are 1/6, 42/253, and 85/512. We must select the convergents that have a denominator smaller than $N = 21$ (since $r < N$)[2]. This method yields 1/6, and then $r = 6$. We check that $2^6 \equiv 1$ modulo 21, and the quantum part of the algorithm ends with the correct answer. The order $r = 6$ is an even number, therefore $\mathrm{GCD}(2^{(6/2)} \pm 1, 21)$ gives two non trivial factors of 21. A straightforward calculation shows that any measured result in the second peak (say $81 \leq j \leq 89$) yields the convergent 1/6.

Consider now the third peak, which corresponds to $k_0/6$, $k_0 = 2$. We apply again the method of continued fraction approximation, which yields 1/3, for any j in the third peak (say $167 \leq j \leq 175$). In this case, we have obtained a factor of r ($r_1 = 3$), since $2^3 \equiv 8 \not\equiv 1$ modulo 21. We run the quantum part of the algorithm again to find the order of 8. We eventually obtain $r_2 = 2$, which yields $r = r_1 r_2 = 3 \times 2 = 6$.

The fourth and fifth peaks yield also factors of r. The last peak is similar to the second, yielding r directly.

The general account of the succeeding probability is as follows. The area under all peaks is approximately the same: ≈ 0.167. The first and fourth peaks have a nature different from the others—they are not spread. To calculate their contribution to the total probability, we take the basis equal to 1. The area under the second, third, fifth, and last peaks are calculated by adding up $\mathrm{Prob}(j)$, for j running around the center of each peak. So, in approximately 17% cases, the algorithm fails (1st peak). In approximately 33% cases, the algorithm returns r in the first round (2nd and 6th peaks). In approximately 50% cases, the algorithm returns r in the second round or more (3rd, 4th, and 5th peaks). Now we calculate the probability of finding r in the second round. For the 3rd and 5th peaks, the remaining factor is $r_2 = 2$. The graph equivalent to Fig. 9 in this case has 2 peaks, then the algorithm returns r_2 in 50% cases. For the 4th peak, the remaining factor is $r = 3$ and the algorithm returns r_2 in 66.6% cases. This amounts to $\frac{2\times 50\% + 66.6\%}{3}$ of 50%, which is equal to around 22%. In summary, the success probability for $x = 2$ is around 55%.

8 Fourier transform in terms of the universal gates

In the previous section, we have shown that Shor's algorithm is an efficient probabilistic algorithm, assuming that the Fourier transform could be implemented efficiently. In this section, we decompose the Fourier transform in terms of the universal gates: CNOT and 1-qubit gates. This decomposition allows one to measure the efficiency of the quantum discrete Fourier transform and shows how to implement it in an actual quantum computer.

[2]The inequality $r \leq \varphi(N)$ follows from the Euler's theorem: $x^{\varphi(N)} \equiv 1 \mod N$, where x is a positive integer coprime to N and φ is the Euler's totient function ($\varphi(N)$ gives the number of positive integers less than N, coprime to N). The inequality $\varphi(N) < N$ follows from the definition of φ. (see [14] page 492)

The Fourier transform of the states of the computational basis is

$$\mathrm{DFT}(|j\rangle) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k/N} \, |k\rangle \, . \tag{25}$$

Noting that the right hand side of Eq. (25) has N terms and the computational basis has N states, we derive that the complexity to calculate classically the Fourier transform of the computational basis using Eq. (25) is $O(N^2) = O(2^{2n})$ – double exponential growth. A very important result in Computer Science was the development of the classical fast Fourier transform (FFT), which reduced the complexity to $O(n2^n)$ [15]. In the present context we show the improvement by recognizing that the rhs of (25) is a very special kind of expansion, which can be fully factored. For example, the Fourier transform of $\{|0\rangle, |1\rangle, |2\rangle, |3\rangle\}$ can be written as

$$\begin{aligned}
\mathrm{DFT}(|0\rangle) &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \\
\mathrm{DFT}(|1\rangle) &= \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + i\,|1\rangle}{\sqrt{2}}\right) \\
\mathrm{DFT}(|2\rangle) &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \\
\mathrm{DFT}(|3\rangle) &= \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - i\,|1\rangle}{\sqrt{2}}\right) .
\end{aligned} \tag{26}$$

Note that in example (26), we are using base 2 in order to factor the rhs. Let us now factor the general expression. The first step is to write (25) in the form

$$\mathrm{DFT}(|j\rangle) = \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \sum_{l=1}^{n} \frac{k_l}{2^l}} \, |k_1\rangle \otimes \ldots \otimes |k_n\rangle \, , \tag{27}$$

where the ket $|k\rangle$ was converted to base 2 and we have used the expansion $k = \sum_{l=1}^{n} k_l 2^{n-l}$ in the exponent. Using that the exponential of a sum is a product of exponentials, (27) turns into a (non-commutative) product of the following kets:

$$\mathrm{DFT}(|j\rangle) = \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \prod_{l=1}^{n} \left(e^{2\pi i j \frac{k_l}{2^l}} \, |k_l\rangle \right) . \tag{28}$$

Now we factor (28) by interchanging the sums and the product:

$$\mathrm{DFT}(|j\rangle) = \frac{1}{\sqrt{2^n}} \prod_{l=1}^{n} \sum_{k_l=0}^{1} \left(e^{2\pi i j \frac{k_l}{2^l}} \, |k_l\rangle \right) . \tag{29}$$

We easily convince ourselves that the last equation is correct by going backwards: simply expand the product in Eq. (29) and then put all sums at the beginning of the resulting expression to obtain (28). Expanding the sum of Eq. (29) and then the product, we

finally get

$$\begin{aligned}\mathrm{DFT}(|j\rangle) &= \frac{1}{\sqrt{2^n}}\prod_{l=1}^{n}\left(|0\rangle + e^{2\pi i j/2^l}\,|1\rangle\right) \\ &= \left(\frac{|0\rangle + e^{2\pi i \frac{j}{2}}\,|1\rangle}{\sqrt{2}}\right)\otimes\left(\frac{|0\rangle + e^{2\pi i \frac{j}{2^2}}\,|1\rangle}{\sqrt{2}}\right)\otimes\ldots\otimes\left(\frac{|0\rangle + e^{2\pi i \frac{j}{2^n}}\,|1\rangle}{\sqrt{2}}\right). \quad (30)\end{aligned}$$

The complexity to calculate Eq. (30) for one $|j\rangle$ is $O(n)$, since there are n terms in the product. The complexity in the classical calculation of the fast Fourier transform of the whole computational basis is still exponential – $O(n2^n)$, since the calculation is performed on each of the 2^n basis elements, one at a time. On the other hand, the quantum computer uses quantum parallelism, and the Fourier transform of the state

$$|\psi\rangle = \sum_{a=0}^{2^n-1} F(a)\,|a\rangle\,,$$

that has an exponential number of terms, is calculated with one application of the quantum Fourier transform. The Fourier transform of the 2^n basis elements is performed simultaneously, so the complexity of the quantum Fourier transform is measured by the size of its circuit. We now show that it requires $O(n^2)$ gates.

Consider the circuit of Fig. 10. It is easy to check that the value of the qubits $|j_m\rangle$, $m \neq l$, does not change. Let us now check the hard one: $|j_l\rangle$. The unitary matrices R_k are defined as

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & \exp\left(2\pi i \frac{1}{2^k}\right) \end{bmatrix}.$$

Each R_k gate is controlled by the qubit $|j_{k+l-1}\rangle$. So, if $j_{k+l-1} = 0$, then R_k must be replaced by the identity matrix (no action), and if $j_{k+l-1} = 1$, then R_k comes in action. This means that, for calculation purposes, the R_k's controlled by $|j_{k+l-1}\rangle$ can be replaced by the 1-qubit gates

$$CR_k = \begin{bmatrix} 1 & 0 \\ 0 & \exp\left(2\pi i \frac{j_{k+l-1}}{2^k}\right) \end{bmatrix}. \quad (31)$$

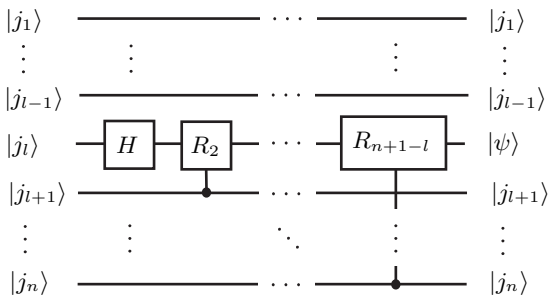

Figure 10: Part of the quantum Fourier transform circuit that acts on qubit $|j_l\rangle$. The value of all qubits does not change, except $|j_l\rangle$ that changes to $|\psi\rangle = \frac{|0\rangle + e^{2\pi i \frac{j}{2^{n+1-l}}}\,|1\rangle}{\sqrt{2}}$.

In order to simplify the calculations, note that

$$H\,|j_l\rangle = \frac{|0\rangle + e^{2\pi i \frac{j_l}{2}}\,|1\rangle}{\sqrt{2}} = CR_1\,|+\rangle\,, \tag{32}$$

where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. So instead of using

$$|\psi\rangle = CR_{n+1-l} \ldots CR_2\; H\,|j_l\rangle\,,$$

which can be read directly from Fig. 10, we will use

$$|\psi\rangle = CR_{n+1-l} \ldots CR_2 CR_1\,|+\rangle\,.$$

We define

$$PR_{n+1-l} = \prod_{k=n+1-l}^{1} CR_k, \tag{33}$$

where the product is in the reverse order. Using (31) and (33), we get

$$\begin{aligned} PR_{n+1-l} &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 0 & \exp 2\pi i\left(\frac{j_n}{2^{n+1-l}} + \ldots + \frac{j_l}{2}\right) \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 0 & \exp\left(2\pi i \frac{j}{2^{n+1-l}}\right) \end{bmatrix}, \end{aligned} \tag{34}$$

where we have used that $j = \sum_{m=1}^{n} j_m 2^{n-m}$ and the fact that the first $l-1$ terms of this expansion do not contribute—they are integer multiples of $2\pi i$ in (34). We finally get

$$\begin{aligned} |\psi\rangle &= PR_{n+1-l}\,|+\rangle \\ &= \frac{|0\rangle + e^{2\pi i \frac{j}{2^{n+1-l}}}\,|1\rangle}{\sqrt{2}}. \end{aligned} \tag{35}$$

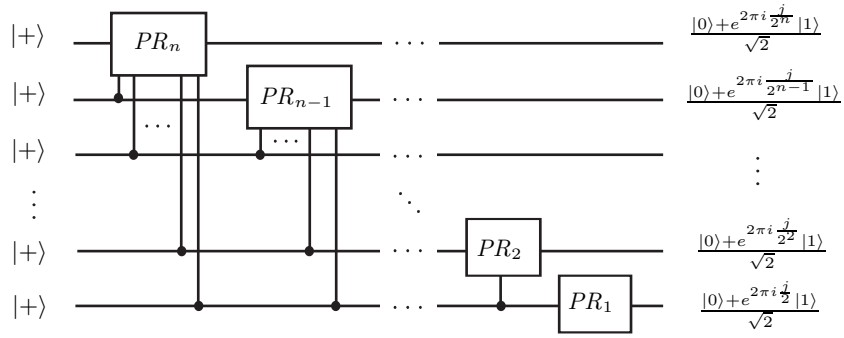

Figure 11: Intermediate circuit for the quantum Fourier Transform. The input is taken as $|+\rangle$ for calculation purposes as explained in Eq. (32). The output is in reverse order with respect to Eq. (30).

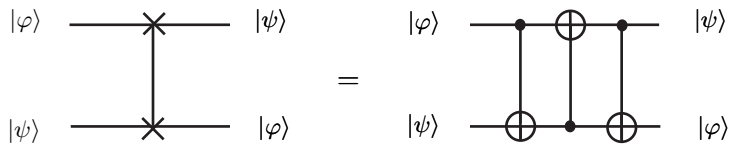

Figure 12: The swap circuit.

Note that PR_{n+1-l} cannot be implemented directly acting only in the l-th qubit because it requires the values of j_{l+1} to j_n.

The next step is the circuit of Fig. 11. We have merged the R_k gates using Eq. (33). The gates PR_k (k from n to 1) are placed in sequence in Fig. 11, so that the output of the first qubit is the last term of Eq. (30), corresponding to the action of PR_n on $|\psi_1\rangle$ controlled by the other qubits, which do not change. The same process is repeated by PR_{n-1} acting on $|\psi_2\rangle$, yielding the term before the last in Eq. (30), and so on, until reproducing all the terms of the Fourier transform. Now it remains to reverse the order of the states of the qubits.

In order to reverse the states of 2 generic qubits, we use the circuit of Fig. 12. Let us show why this circuit works as desired. Take the input $|\varphi\rangle|\psi\rangle = |0\rangle|1\rangle$. The first CNOT of Fig. 12 does not change this state; the upside down CNOT changes to $|1\rangle|1\rangle$; and the last CNOT changes to $|1\rangle|0\rangle$. The output is $|\psi\rangle|\varphi\rangle$. If we repeat the same process with $|0\rangle|0\rangle$, $|1\rangle|0\rangle$, and $|1\rangle|1\rangle$, we conclude that the circuit inverts all states of the computational basis, therefore it inverts a generic state of the form $|\varphi\rangle|\psi\rangle$.

The decomposition is still not complete. It remains to write the controlled R_k gates in terms of CNOT and 1-qubit gates. This decomposition is given in Fig. 13. The verification of this decomposition is straightforward. One simply follows what happens to the computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ in both circuits.

The complete circuit for the quantum Fourier transform is given in Fig. 14. Now we can calculate the complexity of the quantum Fourier circuit. Counting the number of elementary gates in Figs. 10 to 13 we get the leading term $5n^2/2$, which implies that the complexity is $O(n^2)$.

By now one should be asking about the decomposition of V_x in terms of the elementary

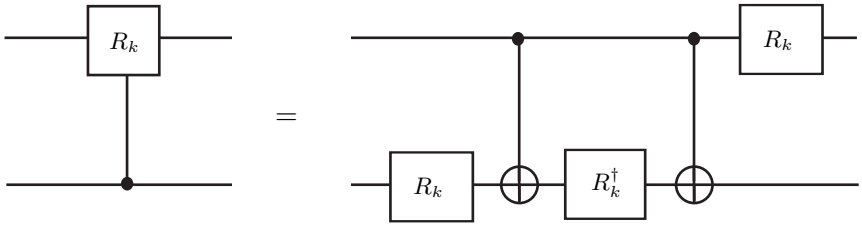

Figure 13: Decomposition of the controlled R_k gates in terms of the universal gates.

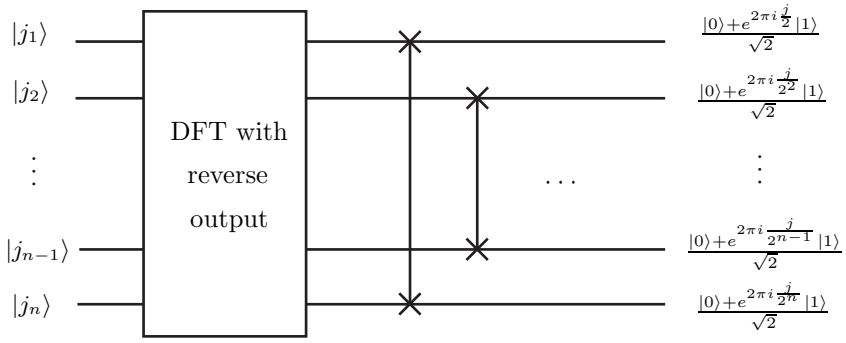

Figure 14: The complete circuit for the quantum Fourier Transform.

gates. V_x is the largest gate of Fig. 5. Actually, Shor stated in his 1997 paper that V_x is the "bottleneck of the quantum factoring algorithm" due to the time and space consumed to perform the modular exponentiation (see [4] page 10). The bottleneck is not so strict though since, by using the well known classical method of repeated squaring and ordinary multiplication algorithms (see [14] page 69), the complexity to calculate modular exponentiation is $O(n^3)$. The quantum circuit can be obtained from the classical circuit by replacing the irreversible classical gates by the reversible quantum counterpart. V_x is a problem in recursive calls of the algorithm when x changes. For each x, a new circuit must be built, what is troublesome at the present stage of hardware development.

Acknowledgments

We thank the Group of Quantum Computation at LNCC for stimulating discussions on the subject.

References

[1] R.P. Feynman, Simulating Physics with computers, Int. J. Theor. Phys. **21** (1982) 467-488.

[2] D. Deutsch and R. Jozsa, Rapid solution of problems by quantum computation, Proc. R. Soc. London **A439** (1992) 553-558.

[3] D. Simon, On the power of quantum computation, Proc. 35th Annual Symposium on Foundations of Computer Science (1994) 116 and SIAM Journal on Computing **26** (1997) 1474-1483.

[4] P. Shor, Algorithms for Quantum Computation: Discrete Logarithm and Factoring, Proc. 35th Annual Symposium on Foundations of Computer Science (1994) 124-134 and SIAM J. Comput. 26 (1997) 1484-1509.

[5] A. Ekert and R. Jozsa, Quantum computation and Shor's factoring algorithm, Reviews of Modern Physics **68** (1996) 733-753.

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

[7] J. Preskill, Quantum Information and Computation, Lecture Notes, California Institute of Technology (1998).

[8] D. Aharonov, Quantum Computation, Annual Reviews of Computational Physics, ed. Dietrich Stauffer, World Scientific, vol. VI (1998).

[9] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, Elementary gates for quantum computation, Phys. Rev. **A52** (1995) 3457-3467.

[10] C.H. Papadimitriou, Computational Complexity, Addison Wesley Pub. Co., Massachussetts (1994).

[11] L.M.K. Vandersypen, M. Steffen, G. Breyta, C.S. Yannoni, M.H. Sherwood, and I.L. Chuang, Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance, Nature, 414 (2001) 883-887.

[12] S.J. Lomonaco, Shor's quantum factoring algorithm, Proceedings of Symposia in Applied Mathematics, Vol. 58, American Mathematical Society (2002) and (quant-ph/0010034).

[13] G. Einarsson, Probability Analysis of a Quantum Computer, quant-ph/0303074 (2003).

[14] J. von zur Gathen and J. Gerhard, Modern Computer Algebra, Cambridge University Press, Cambridge (1999).

[15] J.W. Cooley and J.W. Tukey, An algorithm for machine calculation of complex Fourier series, Math. Comp. **19** (1965) 297-301.