Shor's Algorithm for Period Finding on a Quantum Computer

Peter Young

(Dated: December 5, 2019)

When computers we build become quantum,
Then spies of all factions will want 'em.
Our codes will all fail,
And they'll hack our email,
But crypto that's quantum will daunt 'em.

This is a slightly modified version of a limerick by Peter and Jennifer Shor. (The original version is printed in the book by Nielsen and Chuang.) Continuing in a literary vein, on p. 453 of Nielsen and Chuang is a very well-crafted (Shakespearean) sonnet by Daniel Gottesman on quantum error correction. It seems that quantum computing brings out latent literary qualities in scientists who work on it (except me!).

I. INTRODUCTION

Consider an integer N composed of two prime factors p and q, i.e. $N = p\,q$. In a separate handout *Using Period Finding to Factor an Integer* `https://young.physics.ucsc.edu/150/period.pdf` we show how to determine the factors of N from the period r of the function

$$f(x) \equiv a^x \,(\mod N)\,, \tag{1}$$

where a is some number less than N and which has no factors in common with N. The period is the smallest value $x = r$ such that

$$a^r \,(\mod N) = 1. \tag{2}$$

In 1994 Peter Shor[1] developed a famous quantum algorithm for period finding with which one could factor a large integer much faster than any known algorithm running on a classical computer. The ability to factor a large integer can be used to decode messages sent down a public channel (such as the internet) which have been encrypted with the RSA scheme. The first four lines of the above limerick refer to this[1].

[1] The last line of the limerick refers to quantum key distribution (QKD) which will be discussed separately.

The RSA encryption scheme is describe in the separate handout *RSA (Rivest-Shamir-Adleman) encryption* `https://young.physics.ucsc.edu/150/rsa.pdf`. Let us denote by n_0 the number of bits needed to contain N. In cryptography, N may have of order 600 digits (so $n_0 \sim 2000$ bits).

Here we describe in detail Shor's algorithm to determine the period of the function $f(x)$ in Eq. (1). Useful references are [2–4].

II. MODULAR EXPONENTIATION

In Shor's algorithm the period is found by a Quantum Fourier transform of the function in Eq. (1) evaluated for $x = 0, 1, 2, \cdots, 2^n - 1$. What do we take for n? Now the period may be comparable to N and according to Mermin[2] *Quantum Computer Science* we need at least N periods in the data, i.e. $2^n > N^2$, and so set $n = 2n_0$. We will see why the doubling of the number of qubits is necessary in Sec. V. Hence, if $n_0 \sim 2000$ we have $n \sim 4000$.

It would seem to be a formidable (nay, impossible) task to calculate $a^x \pmod N$ for x from 1 to 2^{4000}. However, it can be done as follows using quantum parallelism. Compute $a, a^2, a^4, \cdots a^{2^n} \pmod N$ by successively squaring. This only takes n multiplications and so is very quickly done on a classical or quantum computer. Let the binary expansion of x be

$$x = x_{n-1}x_{n-2}\cdots x_2 x_1 x_0. \tag{3}$$

Then we have

$$a^x = \prod_{j=0}^{n-1} \left(a^{2^j}\right)^{x_j} . \tag{4}$$

For example for $n = 4$, $x = 10$, the binary expansion of x is 1010 (note the least significant bit is to the right) so

$$a^x = \left(a^8\right)^1 \left(a^4\right)^0 \left(a^2\right)^1 \left(a^1\right)^0 . \tag{5}$$

On a quantum computer each digit of x is represented by one qubit, and, as we shall see, Eq. (4) can be computed efficiently *for all x between 1 and $2^n - 1$* on a quantum computer using quantum parallelism. The use of Eq. (4) to compute a^x for a huge range of x is called "*modular exponentiation*".

A schematic circuit diagram for doing modular exponentiation is shown in Fig. 1. There are n upper or "input" qubits and n_0 lower or "output" qubits, where one usually takes $n = 2n_0$ as discussed above. We will call the *set* of input qubits the "input register", and similarly denote

the output qubits as the "output register". The notation "input" and "output" can be rather confusing. It is used because, after modular exponentiation, the input qubits contain the values of x, and the output qubits contain the function values $f(x)$. Nonetheless the both input and output registers are present at the start of the process (left edge of the circuit diagram in Fig. 1) and at the end (right edge of the circuit).

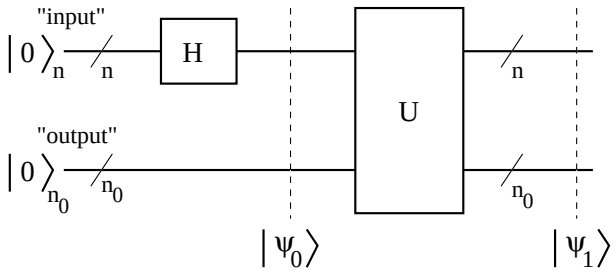

FIG. 1: Schematic circuit diagram for performing the modular exponentiation. The workings of the "black" box U are described in the text. The input state to U is $|\psi_0\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n \, |0\rangle_{n_0}$ and the output from U is $|\psi_1\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n \, |f(x)\rangle_{n_0}$.

Both the input and output qubits are initialized to $|0\rangle$. The input qubits are each run through a Hadamard gate. A Hadamard gate acting on state $|0\rangle$ of one qubit gives the symmetric combination $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Hadamards acting on n qubits gives the symmetric sum of all 2^n basis states. Hence before entering into the box U shown in Fig. 1, the state of the system is

$$|\psi_0\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n \, |0\rangle_{n_0} \, . \tag{6}$$

On exiting the box U, the state of the system has the values of $f(x)$ in the output register, i.e.

$$|\psi_1\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n \, |f(x)\rangle_{n_0} \, . \tag{7}$$

Note that, in general, if the lower (output) qubits were initialized to $|y\rangle$, then after the function acted they would be in state $|y \oplus f(x)\rangle$, but here $y = 0$.

How does the function-evaluating box U work? We start with a certain value for $x \equiv x_{n-1}x_{n-2}\cdots x_2x_1x_0$ in the input register and $1 \equiv 000\cdots001$ in the output register. We also need an additional work register with n_0 qubits, whose contents we will denote by w, with initial value $w = a$. The following steps compute $a^x \pmod N$ using Eq. (4):

- (a) Multiply the output register by w if $x_0 = 1$.
- (b) Replace w by its square $w \to w^2$.

- (a') Repeat (a) but for x_1.
- (b') Repeat (b)
- Continue repeating (a) (with successive bits of x) and (b).

Since all possible values of x occur in the linear superposition in $|\psi_0\rangle$ in Eq. (6), (which is inputted to U) linearity of the operations in U ensures that the state outputted by U is the linear superposition in Eq. (7), with $f(x)$ computed for *all* x.

How many operations does this require? If we consider (b) we need to do n squares of an n_0-bit number. Multiplying two n_0 bit numbers in the simplest way[2] takes $O(n_0^2)$ operations. Since $n = 2n_0$ we see that the operation count for (b) is $O(n^3)$. The operation count for (a) is similar, so the total operation count for modular exponentiation is $O(n^3)$.

III. QUANTUM FOURIER TRANSFORM (QFT)

A schematic of the full circuit for period finding is shown in Fig. 2.

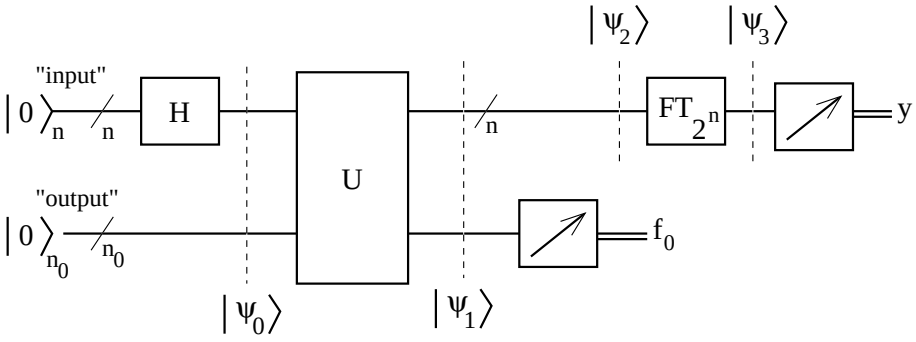

FIG. 2: Schematic circuit diagram for Shor's algorithm for period finding on a quantum computer. The "black" box U does the modular exponentiation as described in the text, see also Fig. 1. The state inputted to U is given by $|\psi_0\rangle$ in Eq. (6) and the state outputted from U is given by $|\psi_1\rangle$ in Eq. (7). A measurement (indicated by the box with the arrow) is performed on the output register, giving some value f_0. The double lines indicate that the measurement gives classical bits which take values 0 *or* 1. The state of the input register is then given by $|\psi_2\rangle$ in Eq. (8), the equally weighted superposition of all values of x for which $f(x) = f_0$. The n input qubits then go through the quantum Fourier transform the result of which is given by $|\psi_3\rangle$ in Eq. (10). A measurement of the input qubits then gives a result y which is close to an integer multiple of $2^n/r$, where r is the period, as discussed in the text.

[2] As mentioned in Nielsen and Chuang [3] *Quantum Computation and Quantum Information*, there are more sophisticated methods of multiplying n-bit numbers which only take $n\,(\ln n)\,(\ln\ln n)$ operations rather than n^2. This gives a total operation count for modular exponentiation of $O(n^2 \ln n\, \ln\ln n)$, hardly more than $O(n^2)$.

The first (left) part of the algorithm is the modular exponentiation also shown in Fig. 1. A measurement is then made of the result in the "output" register from the modular exponentiation routine U. This is indicated by the box with arrow in Fig. 2. The measurement will yield some value for $f(x)$, say f_0. The input register will then contain a superposition of those basis states for which $f(x) = f_0$. Since $f(x)$ is periodic with period r, the possible values of x are of the form $x_0 + kr$, so, after the measurement on the output register, the state of the input register becomes

$$|\psi_2\rangle = \frac{1}{\sqrt{Q}} \sum_{k=0}^{Q-1} |x_0 + kr\rangle_n \,. \tag{8}$$

Here $x_0 < r$, $x_0 + kr < 2^n$ and the number states in the sum is

$$Q = \left[\frac{2^n}{r}\right] , \tag{9}$$

where $[\cdots]$ denotes the integer part. Thus $P_x(x)$, the probability of x, consists of Q delta functions at positions $x_0 + kr, k = 0, 1, \cdots, Q-1$, see Fig. 3.

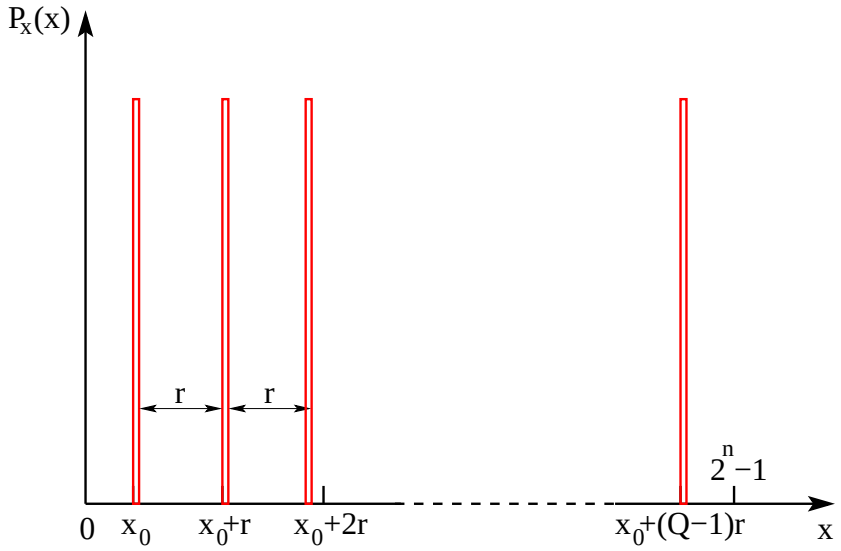

FIG. 3: The probability of getting state x in the input register if a measurement were performed on the input register *before* doing the Quantum Fourier Transform. There are Q delta functions, each with weight $1/Q$ separated by r, the period. The values of x where these delta functions appear, $x_0 + kr, k = 0, 1, \cdots, Q-1$, are those values for which $f(x) = f_0$ the result obtained from the measurement of the output register. A measurement would get a value for $x_0 + kr$ for some k but since we don't know x_0 this is no help in determining the period r. Hence we will need to Fourier transform before measuring the input register.

If we were to measure $|\psi_2\rangle$ we would just get one value of $x_0 + kr$, which, because of the dependence on the unknown quantity x_0, does not give any information from which we might be

able to determine the period r. Therefore, in order to extract information on r, we perform a quantum Fourier transform on the states in Eq. (8):

$$|\psi_3\rangle = \sum_{y=0}^{2^n-1} \left(\frac{1}{\sqrt{2^n Q}} \sum_{k=0}^{Q-1} e^{2\pi i(x_0+kr)y/2^n} |y\rangle_n \right) . \tag{10}$$

The quantum circuit which performs the Quantum Fourier Transform is described in the handout *The Quantum Fourier Transform and a Comparison with the Fast Fourier Transform* `https://young.physics.ucsc.edu/150/QFT-FFT.pdf`. An example for $n = 4$ qubits is shown in Fig. 4 in which the controlled phase gates act on the target qubit according to

$$R_d = \begin{pmatrix} 1 & 0 \\ 0 & e^{\pi i/2^d} \end{pmatrix} , \tag{11}$$

if the control qubit is 1, and otherwise do nothing. Note that R_0 is just the Z gate. Like the controlled Z gate, the controlled phase gate is symmetric between the control and target qubits (the phase is changed only if both qubits are $|1\rangle$), so the control and target qubits can be exchanged. We will use this in Appendix A when we see how to actually eliminate these 2-qubit gates.

Generalizing the diagram in Fig. 4 to the case of n qubits we see that controlled phase gates R_d are required for $d = 1, 2, \cdots, n-1$. Hence, in total, we need n Hadamard gates and $1 + 2 + \cdots + n - 1 = n(n-1)/2$ controlled phase gates. However, as discussed in Sec. 3.9 of Mermin[2], and in Appendix C, it is both impossible to contruct gates giving a phase change which is exponentially small in n, and also not necessary to do this to obtain the QFT with the required precision. Mermin shows that one only needs controlled phase gates R_d for $d < \log_2(\text{const.}\, n)$, where the constant Mermin gives is large but independent of n. Thus the number of controlled phase gates needed *in practice* is of order $n \log_2 n$ which is considerably less than $O(n^2)$ if n is order order several thousand.

In fact we can eliminate the 2-qubit controlled phase gates by measuring each qubit immediately after the gates of the QFT have acted on it, rather than after completion of the QFT. This is discussed in Appendix A.

After the quantum Fourier transform we measure the input register, see Fig. 2, obtaining a value for y. The probability of getting a particular state y is given by the square of the absolute value of a term in the brackets in Eq. (10), i.e.

$$P(y) = \frac{1}{2^n Q} \left| \sum_{k=0}^{Q-1} e^{2\pi i k r y/2^n} \right|^2 . \tag{12}$$

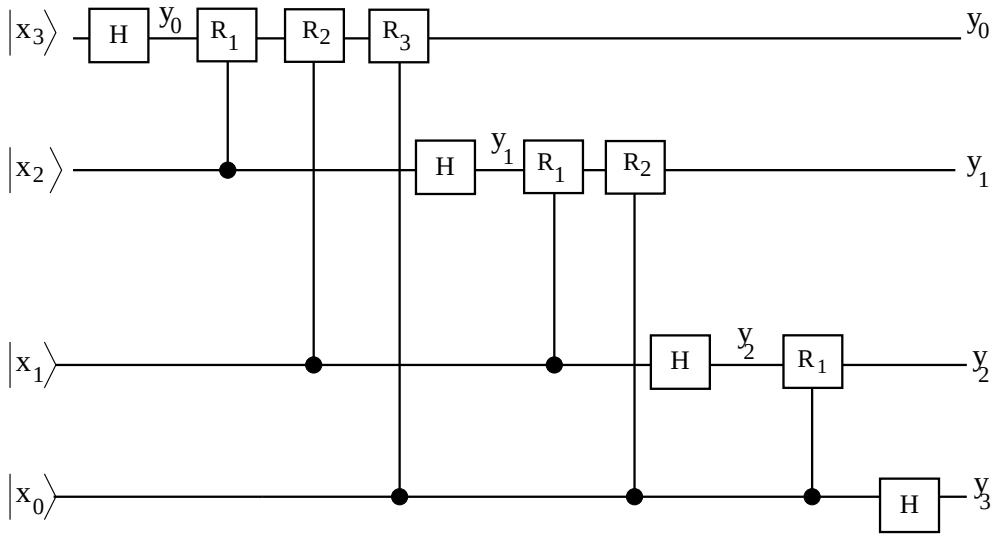

FIG. 4: The circuit for the quantum Fourier transform for $n = 4$ qubits. The controlled phase gates act on the target qubit according to Eq. (11) if the control qubit is 1 and otherwise does nothing. The final swap gates to reverse the order of the qubits outputted on the right are not included here. Note that the controlled phase gate between qubits x_i and x_j is $R_{|i-j|}$ which makes the structure of the circuit quite simple to understand.

Note that the dependence on x_0, which was troublesome before doing the Fourier transform, and appears just as a phase factor in the Fourier transform, Eq. (10), now drops out completely when we take the square of the absolute value to get the probabilities in Eq. (12).

If y could take real values, the exponentials would add up precisely in phase (and so there would be a peak in the probability for y), when $yr/2^n$ is an integer, i.e. for $y = y_m$ where

$$y_m = m\,\frac{2^n}{r}, \tag{13}$$

in which m is an integer. Note that there are r values of m, from 0 to $r-1$ since y runs over a range of 2^n values. We emphasize that y_m is not an integer in general, but the measured values of y *are* are integers, so there will be peaks in $P(y)$ at integer values *close to* the y_m in Eq. (13). See the sketch in Fig. 5. The precise values of $P(y)$ will be calculated in Sec. V.

Hence there is a high probability that we will obtain an integer close to an integral multiple of $2^n/r$.

To summarize this part, $P(y)$ has r peaks separated by $2^n/r$ (the nearest integer below this is Q), see Fig. 5. Note that r, the number of peaks after the QFT (Fig. 5), is equal to the separation between the peaks before the QFT (Fig. 3) and the separation between the peaks after the QFT (essentially Q) is equal to the number of peaks before the QFT. This interchange of the number of

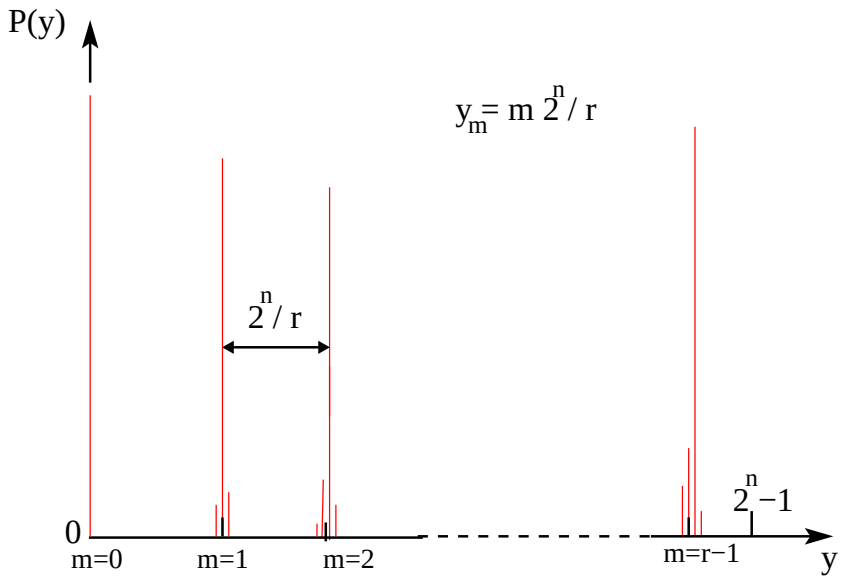

FIG. 5: The probability of getting state y in the input register *after* the Quantum Fourier Transform. There are r peaks at $y_m = m\,2^n/r$ for $m = 0, 1, 2, \cdots, r-1$. Note that $[2^n/r] = Q$ so the *separation* between the peaks in $P(y)$ is, no more than 1 away from Q, the *number* of peaks in the distribution $P_x(x)$ for the state *before* the quantum Fourier transform, see Fig. 3. This is just a sketch; the precise values of $P(y)$ will be calculated in Sec. V, and real data will be shown in Fig. 8.

peaks and their separation on doing the QFT is characteristic of the discrete Fourier Transform.

IV. A SPECIAL CASE: THE PERIOD r IS A POWER OF 2.

In some special cases the period r will be a power of 2. An example discussed by Mermin[2] is if both p and q are both primes of the form $2^\ell + 1$ (e.g. the commonly studied case of $N = 15$). When the period is a power of 2, the range of x (i.e. 2^n) will contain an *exact* integer number of periods. In this situation we will not need n to be as big as $2n_0$ (where n_0 is the number of bits needed to contain N). Rather, we will see that we just need 2^n to be big enough to contain some integer number of periods for us to exactly determine an integer multiple of $2^n/r$.

Here we go through this special case because the mathematics is simpler than the generic case which we will study in the next section.

First of all we check for $N = 15$ that the period *is* a power of 2. Let's take $a = 7$ which has no

factors in common with 15:

$$x = 1, \quad a^x = 7\,, \tag{14a}$$

$$x = 2, \quad a^x = 7 \times 7 = 49 \equiv 4 \ (\text{ mod } 15)\,, \tag{14b}$$

$$x = 3, \quad a^x = 7 \times 4 = 28 \equiv 13 \ (\text{ mod } 15)\,, \tag{14c}$$

$$x = 4, \quad a^x = 7 \times 13 = 91 \equiv 1 \ (\text{ mod } 15)\,, \tag{14d}$$

so the period is $r = 4$, i.e. a power of 2 as claimed.

Now, we perform the sum in Eq. (12). We take any n such that 2^n is a multiple of r. The value of Q is given exactly by

$$Q = \frac{2^n}{r}\,, \tag{15}$$

so Eq. (12) becomes

$$\begin{aligned} P(y) &= \frac{1}{2^n Q} \left| \sum_{k=0}^{Q-1} e^{2\pi i k r y/2^n} \right|^2 , \\ &= \frac{1}{r} \left| \frac{1}{Q} \sum_{k=0}^{Q-1} e^{2\pi i k y/Q} \right|^2 . \end{aligned} \tag{16}$$

Firstly suppose that $y = mQ$ for integer m. It is trivial to see that all the exponentials in Eq. (16) are unity so

$$P(y = mQ) = \frac{1}{r}. \tag{17}$$

Note that there are r distinct values of m, $m = 0, 1, 2, \cdots, r-1$ since y runs over a range of 2^n values and $Q = 2^n/r$, see Eq. (15). Hence the sum of the probabilities for these values of y is unity. Since the *total* probability must be unity there can be no probability for other values of y, as we will now verify.

The sum in Eq. (16) is a geometric series, and for $y \neq mQ$ it can be summed to give

$$\sum_{k=0}^{Q-1} e^{2\pi i k y/Q} = \frac{1 - e^{2\pi i y}}{1 - e^{2\pi i y/Q}}\,. \tag{18}$$

The numerator is zero for all y (recall that y is an integer), but for $y \neq mQ$ the denominator is non-zero, so

$$P(y \neq mQ) = 0\,, \tag{19}$$

as required. Thus, with probability 1, the measured value of y is an integer multiple of $2^n/r$. This is shown in Fig. 6. Superficially, this may look similar to the situation before the QFT shown

in Fig. 3. The difference is that the known quantity x_0 does not appear in Fig. 6 and the delta functions occur at positions y_m where $y_m/2^n = m/r$ from which one can determine r. By contrast, the delta functions in Fig. 3 are at $x_k = x_0 + kr$ from which one can not determine r because one has no idea of the value of x_0.

Notice the reciprocal relation between the period r in the original data in Fig. 3 and the period in the Fourier transformed data which is the size of the dataset, 2^n, *divided* by r. To use terminology from sound waves and frequencies, quite generally, if an original dataset is periodic, the Fourier transform will have a peak at the "fundamental frequency", in this case $2^n/r$, and in addition can have peaks at "higher harmonics" (in this case $m\,2^n/r$ for $m > 1$), and can also have a component at zero "frequency" ($y = 0$ here) if the average of the original data is non-zero. The special nature of the original dataset here (uniformly spaced delta functions, see Fig. 3) leads to the Fourier transform having *equal* weight in all the Fourier components.

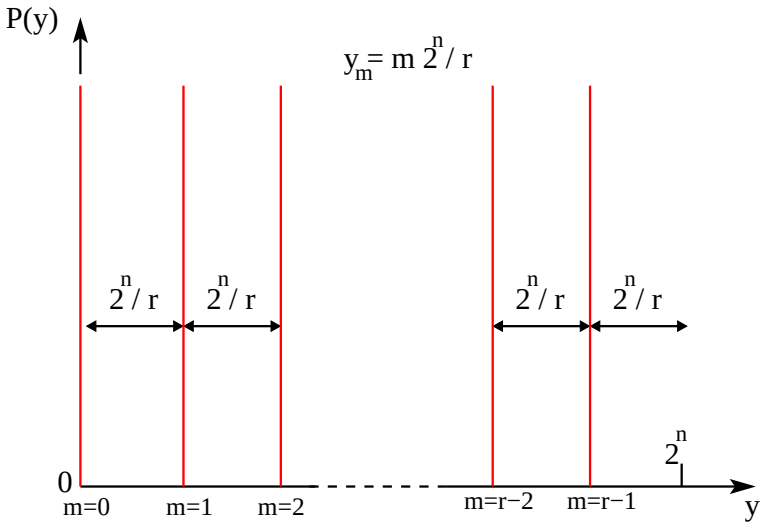

FIG. 6: The probability of getting state y in the input register after the Quantum Fourier Transform for the special case where r is a power of 2 so there are an exact number of periods in the interval 2^m. There are $Q = 2^m/r$ delta functions of equal weight at exactly $y_m = m\,2^n/r$, for $m = 0, 1, \cdots, Q-1$.

Let us give a simple example so we can see in detail how to extract the period r from this knowledge. We take our previous example of $N = 15, a = 7$, for which we found in Eq. (14) that the period is $r = 4$. This means that $7^4 \equiv 1 \mod 15$. We will assume that we have $n = 5$ qubits, so $2^n = 32$. The only possible results of a measurement of y are an integer multiple of $Q = 2^n/r$, so here we have $y = 0, 8, 16$ and 24, each with equal probability 1/4, see Table I.

From the measurement of y we determine the numerator and denominator of the fraction $y/2^n (= m/r)$ for some m, but with any common factor c, shown in the last column of Table I, divided

y	m	$\frac{y}{2^n}\left(=\frac{m_0}{r_0}\right)$	$c=\frac{r}{r_0}$
0	0	0	–
8	1	1/4	1
16	2	1/2	2
24	3	3/4	1

TABLE I: The possible results of a measurement of y for the case of $N = 15, a = 7, 2^n = 32$ for which $r = 4$. A measurement gives both the numerator and denominator of the fraction $y/2^n (= m/r)$ for some m, but with any common factor c, shown in the last column, divided out, so we write $y/2^n$ as m_0/r_0 with $m = cm_0, r = cr_0$. Hence we obtain r_0 (and m_0), but not c. We determine c by trial and error, as discussed in the text.

out, so we write $y/2^n$ as m_0/r_0 with $m = cm_0, r = cr_0$. To determine $r = cr_0$, where c is generally a small integer, we compute $a^{cr_0} \mod N$ for the first few values of $c = 1, 2, \cdots$ and see for what value of c we obtain 1, the result if $cr_0 = r$, see Eq. (2). The common ratio c is unlikely to be large. For example if m is odd, which occurs with probability 1/2, then $c = 1$. Similarly there is probability 1/4 that m is even but not a multiple of 4 in which case c cannot be greater than 2. Proceeding in this vein we see that it is very unlikely that c is large. In the rare case that the common ratio c *is* large, we would stop after the first few values of c and restart the quantum computation (the steps shown in Fig. 2).

In Table I we see that the value $y = 0$ does not give useful information but, since the number of possible results is equal to r and each result is equally probable, the probability of getting $y = 0$ is small if the period r is large (the usual situation if one needs a quantum computer).

In this section, we have seen that in the rare situation that the period is a power of 2, the measurement of y gives an integer multiple of $2^n/r$ *with probability one.* Hence $y/2^n = m/r$ with integer m *exactly* and there is no need to use the continued fraction method described in Appendix B to determine m/r.

However, in the general case, which we discuss in the next section, we firstly need to have $n = 2n_0$ so there are at least N periods in the range of x, and secondly the measurement of y will give, with a probability which is high but less than one, a value such that $y/2^n$ is *close to* (but not equal to) m/r. The continued fraction method in Appendix B is then needed to determine m/r.

V. THE GENERIC CASE: THE PERIOD IS NOT A POWER OF 2.

We now evaluate the sum in Eq. (12) for the generic case when r is not a power of 2 so we do not have an exact integer number of periods in the range of x-values, 2^n, over which $f(x)$ is calculated. As discussed after Eq. (12), $P(y)$ has r peaks in the vicinity of y_m given by Eq. (13). We set

$$\begin{aligned} y &= y_m + \delta_m \,, \\ &= m\frac{2^n}{r} + \delta_m \,, \end{aligned} \tag{20}$$

We will assume that δ_m is small, so we are close to the m-th peak, but $2^n, r$ and m are large, since we only need the quantum algorithm when these numbers are large. (Recall that y, the measured value is an integer, whereas y_m and δ_m are not.)

Equation (12) involves a geometric series which can be summed as follows:

$$\begin{aligned} \sum_{k=0}^{Q-1} e^{2\pi i k r y/2^n} &= \sum_{k=0}^{Q-1} e^{2\pi i k m} e^{2\pi i k r \delta_m/2^n} \,, \\ &= \sum_{k=0}^{Q-1} e^{2\pi i k r \delta_m/2^n} \,, \\ &= \frac{1 - e^{2\pi i Q r \delta_m/2^n}}{1 - e^{2\pi i r \delta_m/2^n}} \,, \\ &= \frac{e^{\pi i Q r \delta_m/2^n} \sin\left(\pi Q r \delta_m/2^n\right)}{e^{\pi i r \delta_m/2^n} \sin\left(\pi r \delta_m/2^n\right)} \,. \end{aligned} \tag{21}$$

Inserting Eq. (21) into Eq. (12) the phase factors drop out and we get

$$P(y) = \frac{1}{2^n Q} \frac{\sin^2\left(\pi Q r \delta_m/2^n\right)}{\sin^2\left(\pi r \delta_m/2^n\right)} \,. \tag{22}$$

Now Q is within an integer of $2^n/r$ and Q is also large so so we can replace $Qr/2^n$ by 1 with negligible error. Also $r/2^n$ is very small, since we take n to be big enough that there are many periods within the range of x computed, so the sine in the denominator can be replaced by its argument. Hence we get, to a good approximation,

$$\boxed{P(y) = \frac{1}{r}\left(\frac{\sin \pi\delta_m}{\pi\delta_m}\right)^2 \,,} \tag{23}$$

for y in the vicinity of y_m. Recall that the relation between δ_m and y is given in Eq. (20). The function in Eq. (23) is plotted in Fig. 7. The area under the curve is 1, and most of the weight is in the peak centered at 0.

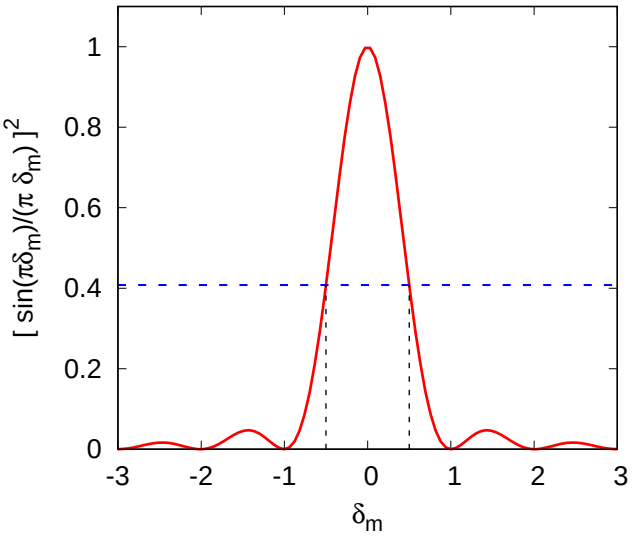

FIG. 7: A plot of the function in Eq. (23), neglecting the factor of $1/r$ which is the number of peaks. The possible values of δ_m are separated by integers, e.g. $\cdots, -1.7, -0.7, 0.3, 1.3, \cdots$. An example of real data is shown in Fig. 9. One of these values must be within $1/2$ of 0 and the figure shows that the probability for this is greater than $4/\pi^2$, the dashed horizontal line. (The dashed vertical lines are at $\delta_m = \pm 1/2$).

To find the period we would like to get the integer y which is *closest* to $m\,2^n/r$ for some integer m i.e. $|\delta_m| < 1/2$. Writing $\pi\delta_m = x$, this corresponds to $|x| < \pi/2$, and in this region

$$\frac{\sin x}{x} > \frac{2}{\pi}\,, \tag{24}$$

so, according to Eq. (23), the probability of getting the nearest integer to y_m is greater than

$$\frac{1}{r}\,\frac{4}{\pi^2} \simeq \frac{0.40}{r}, \tag{25}$$

see Fig. 7. There are r distinct values[3] of m so the total probability of getting the closest integer to *one* of the y_m is greater than 40%.[4]

So, with fairly high probability, we have obtained the nearest integer to $m\,2^n/r$ for some integer m (which we don't know). How can we determine r from this information? We need some post-processing which will be done on a *classical* computer.

[3] One of these is for $m = 0$ which doesn't give useful information but since we are interested in situations where r is large, the difference between r and $r - 1$ is negligible.

[4] In fact, according to Mermin[2], Appendix L, when N is the product of two primes (as we have here) the period is not only less than N but less than $N/2$. As a result, still using $n = 2n_0$ qubits in the input register, the algorithm will provide a divisor of r not only if the measured value of y is the closest integer to $m\,2^n/r$, but also if it is the second, third or fourth closest. This increases the probability of a successful run to about 0.9.

In deriving Eq. (23) we just needed that the range of x studied contains many periods, i.e. $2^n \gg r$. Since r can not be bigger than N we needed $2^n \gg N$. However, to actually extract r we need a stronger condition, $2^n > N^2$, as we shall now see.

We assume now that we have been successful and found a y which is within $1/2$ of $2^n m/r$. Dividing by 2^n we have

$$\left|\frac{y}{2^n} - \frac{m}{r}\right| < \frac{1}{2^{n+1}}\,, \tag{26}$$

so our estimate for m/r is off by no more than $1/(2\,2^n)$. Now any two distinct fractions with denominators less than N must differ[5] by at least $1/N^2$. It follows that the value of m/r can then be obtained using continued fractions, which are discussed in Appendix B. As shown in Theorem A4.16 in Appendix 4 of Ref. [3], if $2^n > N^2$, the ratio m/r will appear as one of the partial sums in the continued fraction representation of y_m, with a denominator less than N (since $r < N$). Hence m/r is equal to the continued fraction representation of $y/2^n$ with the largest denominator less than N. If m and r have a common factor, c say, then the continued fraction representation will divide this out and give m_0/r_0 where $m_0 = m/c, r_0 = r/c$. Thus we actually get r_0 which is a divisor of r.

However, we may be lucky and still get r straight away. As shown in Appendix J of Mermin[2], the probability that two large numbers chosen at random have no common factors is greater than $1/2$. Thus, with probability greater than $1/2$, we get r directly. We can check if r_0 is the period r by computing, on a classical computer, $a^{r_0}(\mod N)$ and seeing if we get 1. If we do not, we would try simple multiples, $r = 2r_0, 3r_0, 4r_0, \cdots$, since it is very unlikely that the common factor is large. If we *are* very unlucky, and the common factor *is* large, we could start again from the beginning, get another value for m/r and hence get another value for r_0, and compute $a^{r_0}(\mod N)$. If this is not 1, then again we try $r = 2r_0, 3r_0, 4r_0, \cdots$. There is also a chance that the measured value of y is not close enough to one of the y_m to get the period from continued fractions. Again, if this happens we need to repeat the whole procedure. However, we will not have to repeat very many times because the probability of success in one run is quite high.

The probabilistic nature of Shor's algorithm, with the resultant need to run the algorithm several times (usually not very many), is a quite common feature of quantum algorithms.

---

[5] Note that

$$\frac{p}{q} - \frac{r}{s} = \frac{ps - rq}{qs} \geq \frac{1}{qs}$$

for p, q, r, s integer, unless the fractions are identical.

VI. AN EXAMPLE

The last section was, perhaps, hard going, so we will try to clarify things by going through a simple example. Consider the following, which was also discussed in the handout *Using Period Finding to Factor an Integer* `https://young.physics.ucsc.edu/150/period.pdf`, $N = 91, a = 4$. The period is $r = 6$ since

$$x = 1, \quad a^x = 4\,, \tag{27a}$$

$$x = 2, \quad a^x = 16\,, \tag{27b}$$

$$x = 3, \quad a^x = 64\,, \tag{27c}$$

$$x = 4, \quad a^x = 64 \times 4 = 256 = 2 \times 91 + 74 \equiv 74 \ (\text{ mod } 91)\,, \tag{27d}$$

$$x = 5, \quad a^x \equiv 74 \times 4 = 296 = 3 \times 91 + 23 \equiv 23 \ (\text{ mod } 91)\,, \tag{27e}$$

$$x = 6, \quad a^x \equiv 23 \times 4 = 92 = 91 + 1 \equiv 1 \ (\text{ mod } 91)\,. \tag{27f}$$

Since the period is not a power of 2 this is a generic example, as discussed in the previous two sections.

order (m)	peak position ($y_m = m\,2^n/r$)	nearest integer	P(nearest int.)
0	0	0	0.167
1	2730.67	2731	0.114
2	5461.33	5461	0.114
3	8192	8192	0.167
4	10922.67	10923	0.114
5	13653.33	13653	0.114

TABLE II: The peak positions in the Fourier transform for the example discussed in this handout. The output is at integer values of y and the nearest integers to the peaks are shown along with the probability at those nearest integer values. Neglecting the zeroth order peak at $y = 0$, which doesn't give useful information, the sum of the other probabilities at the nearest integers is 0.623, so we have a greater than 60% probability of obtaining the nearest integer to a non-zero multiple of $2^n/r$, from which one can deduce r using continued fractions, as discussed in the text and Appendix B.

One needs $n_0 = 7$ bits to represent N so we take $n = 2n_0 = 14$. Hence

$$\frac{2^n}{r} = 2730.67 \tag{28}$$

so

$$Q = 2730\,. \tag{29}$$

Hence there are 2730 (and two thirds) periods in our data. As discussed in Mermin[2] and Sec. V we need at least $N\,(=91)$ periods so 2730 is something of an overkill. The peaks in the Fourier transform, which are at integers next to multiples of $2^n/r$ as discussed above, are shown in Table II.

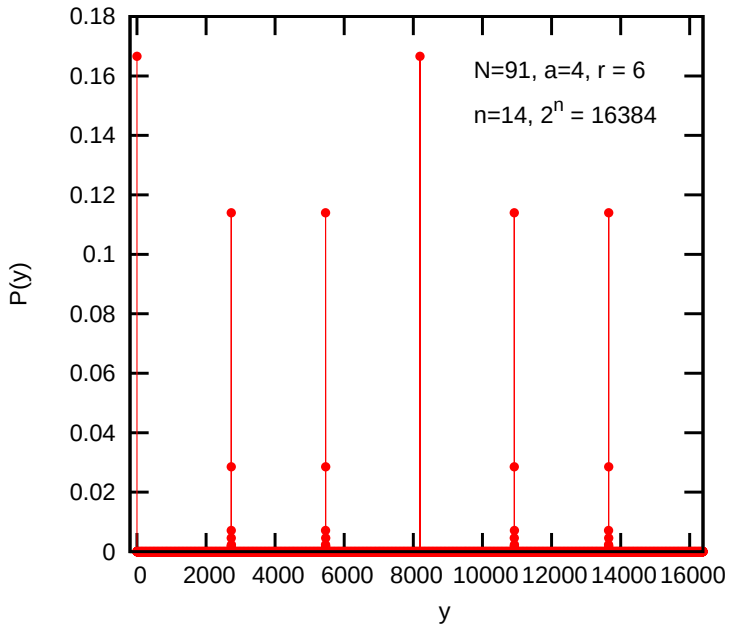

FIG. 8: Probabilities for the different components of the Fourier transformed state for the example studied with $N = 91, a = 4$ for which the period is $r = 6$. There are six sharp peaks near $y_m = m\,2^n/r$, for $m = 0, 1, \cdots, 5$. The one at $y = 0\,(m = 0)$ doesn't give useful information. However, the probability of hitting the highest point of one of the other five peaks, i.e. the nearest integer to a non-zero multiple of $2^n/r$, is greater than 60%, see Table II. If the measurement gives one of these results, it can then be used to determine the period r, as discussed in Appendix B. Note that the number of peaks, 6 here, is equal to r, the period of $f(x)$, which we want to calculate. However, this observation does not help us determine r since a measurement just gives a single value of y. The *separation* between the peaks is $2^n/r$, which is not an integer but the integer below it is Q, given by Eq. (9). Q is also equal to the *number* of values of x with a non-zero amplitude in the state before Fourier Transforming, see the sketch in Fig. 3. A blowup of the $m = 2$ peak is shown in Fig. 9.

I have evaluated $P(y)$ numerically from Eq. (12) and the results are shown in Fig. 8. There are $r = 6$ peaks at values close to $y_m = m\,2^2/r$. There is a trivial peak at exactly $y = 0\,(m = 0)$ but this can not give any useful information about the period r. For the other 5 peaks, the peaks are not, in general, centered at exactly integer values, so the possible observed (integer) values of y are a set of discrete values around each peak, as shown in the histogram in Fig. 9 which blows up the region around the $m = 2$ peak.

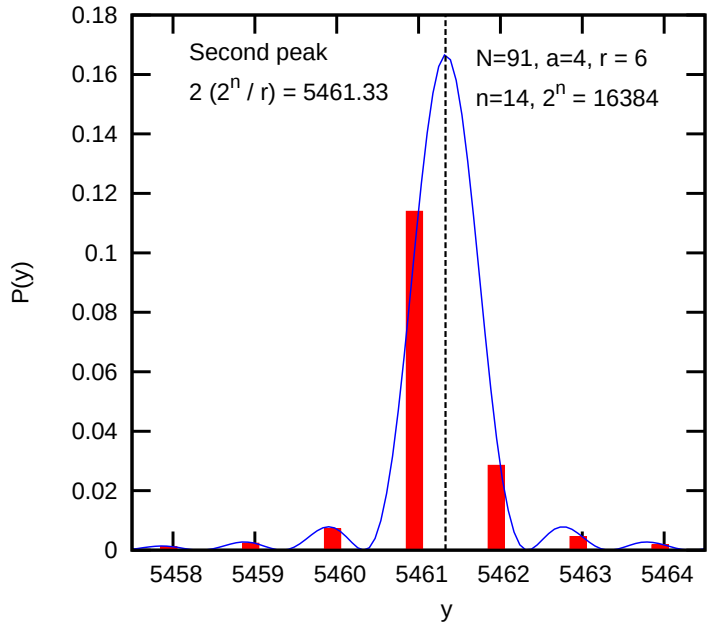

FIG. 9: A blowup of the region around the $m = 2$ peak in Fig. 8 (see also Table II). The histogram is obtained from numerical evaluation of Eq. (12). The probability is dominated by the biggest bar, which is at $y = 5461$ the nearest integer to $y_2 = 2 \times (2^n/r) = 5461.33$ (indicated by the vertical dashed line). The solid curve is the expression shown in Eq. (23) (with y considered to be a continuous variable).

As discussed in Sec. V, the sum in Eq. (12) can be evaluated, and is given, to a good approximation, by Eq. (23) in the region of the m-th peak, where y is given by Eq. (20), and y_m, given by Eq. (13), indicates the peak position. (Recall that y itself is an integer.) The function in Eq. (23) is plotted for continuous y as the solid curve in Fig. 9. When evaluated at integer y, it agrees very well with the values numerically computed from Eq. (12) which are shown as the histogram in Fig. 9.

Note that δ_m in Eq. (23) is defined in Eq. (20) and can be written as

$$\delta_m = \epsilon + \ell \tag{30}$$

where ℓ is an integer and $|\epsilon| < 0.5$. Note too that

$$\sum_{\ell=-\infty}^{\infty} \left(\frac{\sin(\pi(\epsilon+\ell))}{\pi(\epsilon+\ell)} \right)^2 = 1, \tag{31}$$

for arbitrary ϵ (recent versions of Mathematica know this). Hence, according to Eqs. (23), (30), and (31), the weight around each of the peaks in Fig. 8 is equal to $1/r$ ($= 1/6$ here). There are r peaks so the total probability is $r \times (1/r) = 1$ as required. Referring to Fig. 9, the weight in the largest bar is 0.114 which is 68% of 1/6, the total weight in all the bars for this peak.

From Table II we see that the probability of getting the nearest integer to an integral multiple of $2^n/r$ is greater than 60%. Let's suppose we get one of these. In fact, lets suppose we get the large bar at $y = 5461$ in Fig. 9. (Recall that Fig. 9 is a blowup of the $m = 2$ peak in Fig. 8.) Given the measured value, $y = 5461$, we will now see how to determine the period r using continued fractions which are described in Appendix B. We define $x = y/2^n$. This is close to m/r, where r, the period, is what we want to determine.

Since r is no greater than N, as discussed in Sec. V, the best guess for x is the partial sum having the largest denominator less than N. As stated above we assume measurement gives the value $y = 5461$, the highest histogram for the peak in Fig. 9. Then we determine the continued fraction representation for $x = 5461/16384$ (since $n = 14$ we have $2^n = 16384$). Since this is a rational fraction the continued fraction terminates and has only the following denominators

$$c_0 = 0,\ c_1 = 3,\ c_2 = 5461. \tag{32}$$

The partial sums are 1/3 and 5461/16384. The latter has a denominator bigger than N (= 91) so we neglect it and conclude that[6]

$$\frac{m}{r} = \frac{1}{3}\,. \tag{33}$$

It is possible that m and r have a common factor, i.e. $m = c, r = 3c$ for some integer c. We try some small values for c. Starting with $c = 1$, so $r = 3$, we compute $a^3 \pmod{91}$ and find that it is not 1, see Eq. (27c). However, we find that $c = 2$ does work, since $a^6 \equiv 1 \pmod{91}$, see Eq. (27f). Hence the period r is equal to 6, the desired result.

VII. SUMMARY

What is the operation count for Shor's period finding algorithm?

To factor an integer with n bits, the QFT requires, in principle, $O(n^2)$ operations, as shown in section III. Note, however, as discussed there, in Appendix C, and in Mermin [2], in practice one only needs of order $n \log_2 n$ gates to do the QFT to the necessary precision.

The computation of the function values using modular exponentiation takes $O(n^3)$ operations, as shown in section II (but see footnote 2 on page 4 which states that the operation count is

[6] In this case, where there are many more than N periods in the intervals 2^n, one gets the right answer if the measurement gives one of the other nearby y values. For example, if we get $y = 5760$ (the third closest to the peak), the continued fraction coefficients are $0, 3$, and 1365. We do not include 1365 because it gives a denominator greater than N, so we again get $m/r = 1/3$.

$O(n^2 \log n \log\log n)$, not much more than $O(n^2)$, if one uses a sophisticated method for multiplying two large numbers).

What about the continued fraction part, which is, of course, done on a classical computer? Each division of an n-bit number takes of order n^2 operations if the division is done in a simple way. In fact, division can be rewritten as several multiplications, see `https://en.wikipedia.org/wiki/Division_algorithm`, so the operation count can be reduced to that for multiplication, i.e. $O(n \log n \log\log n)$. The depth of the continued fraction where the denominator is $O(N)$ is $O(\log N)$, since the coefficients in the continued fraction multiply to get the numerator and denominator. This is $O(n)$ since N contains no more than $n/2$ bits. Hence the operation count for the continued fraction post-processing is $O(n^3)$, but recall that this is done on a classical computer. Again the count is not much more than $O(n^2)$ if one uses a sophisticated method for dividing two large numbers.

Hence, the overall operation count of Shor's algorithm is $O(n^3)$, which can be reduced to $O(n^2 \log n \log\log n)$ using sophisticated methods for multiplying and dividing large numbers.

As we discussed, from period finding we can factor a large integer N. Shor's algorithm for factoring integers therefore runs in polynomial time as a function of n, the number of bits in N. For comparison, no polynomial time classical algorithm for factoring integers is known. The fastest known classical algorithm, the general number field sieve (GNFS), takes a time $\exp(\text{const.}\, n^{1/3} \log^{2/3} n)$. It is currently not known whether a polynomial time classical algorithm exists or not.

Even though the power of n in the exponent of the GNFS algorithm is less than one, it still much slower for large n than Shor's polynomial-time algorithm. Hence, if the considerable technical difficulties could be overcome, and a quantum computer with a sufficiently large number of qubits built with the error rate made sufficiently low, then such a device could decode encrypted messages currently being sent down the internet, and which are currently impossible to decode on a classical computer.

Appendix A: Eliminating the two-qubit gates

It is possible to replace the 2-qubit gates by 1-qubit gates which act or not depending on the result of a measurement. This is important from a technological point of view since 1-qubit gates are much easier to implement than 2-qubit gates. The point is that we measure the final state of the QFT anyway, and we will see that we can eliminate the 2-qubit gates by measuring each qubit *immediately all the gates of the QFT have acted on it* rather than waiting until the QFT is

completed. We now see how to do this.

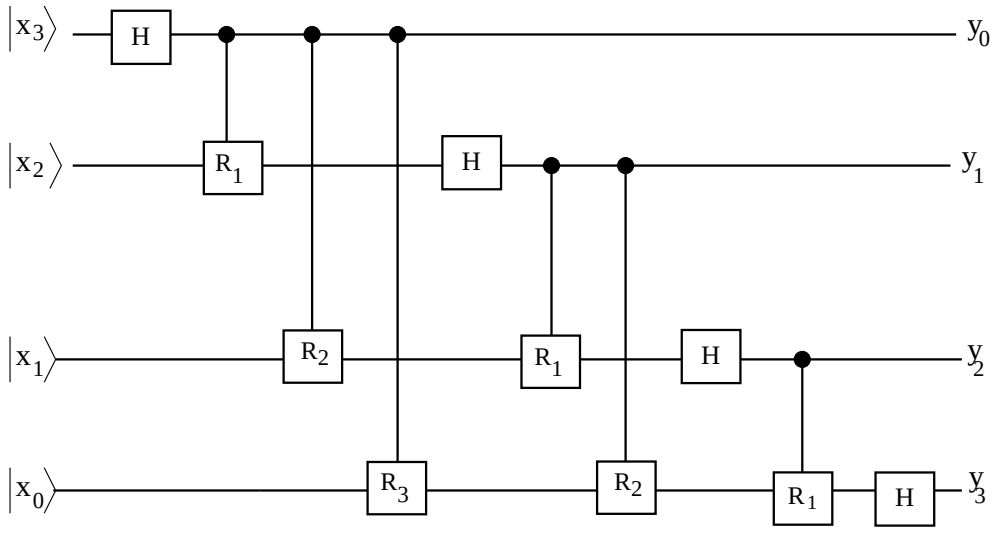

FIG. 10: Circuit equivalent to Fig. 4 but with the target and control qubits interchanged on the controlled phase gates.

First of all we note that, similar to the control-Z gate, the target and control qubits in the controlled phase gates can be interchanged. Hence Fig. 4 is equivalent to Fig. 10.

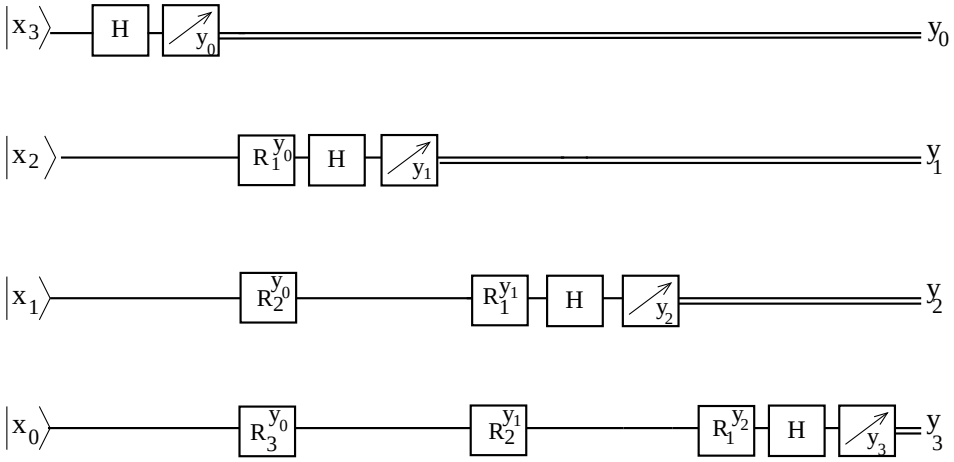

FIG. 11: Circuit for the QFT with 4 qubits equivalent to Fig. 10 but in which each qubit is measured immediately after the Hadamard gate. Subsequent phase gates (on qubits lower in the diagram) are controlled by classical circuits (not shown) which use the values of the already measured qubits. Note that $R_1^{y_0}$ means R_1 to the power y_0. Since y_0 is 0 or 1 this gives R_1 if $y_0 = 1$ and 1 if $y_0 = 0$. Hence we obtain the required control, but done by a classical circuit rather than the 2-qubit controlled phase gates in Fig. 10.

In Fig 10 we see that, for each qubit, once the Hadamard and the phase gates have acted nothing else happens so the qubit could be measured at this point. (Recall that time flows from

left to right in circuit diagrams). Consider the top qubit x_3 which, on output, is y_0. We measure it immediately after the Hadamard. If the result is $y_0 = 1$ then the R_1 phase gate for x_2 is activated, as well as the R_2 phase gate for x_1 and the R_3 phase gate for x_0. However, if the result is $y_0 = 0$ then those phase gates are not activated. Since y_0 has been measured, this control can be done by a *classical* circuit, which is much simpler to implement than a 2-qubit quantum gate. Similarly we measure x_2, which is y_1 on output, immediately after its Hadamard. Hence the R_1 gate on x_1 and the R_2 gate on x_0 can be activated classically if $y_1 = 1$. We can proceed in this way for the whole circuit, measuring the qubit after the Hadamard, and using the result to phase change other qubits, or not, using classical control. The circuit is shown in Fig. 11.

Appendix B: Continued Fractions

Continued fractions are a convenient way of finding a simple rational approximation to a number.

The continued fraction representation of a number x is obtained as follows. If there is an integer part of x call this c_0. Subtract c_0 from x and call the inverse of the remainder x_1, so

$$x = c_0 + \frac{1}{x_1}. \tag{B1}$$

Let the integer part of x_1 be c_1. Subtract c_1 from x_1 and call the inverse of the remainder x_2. Continuing in the same way for c_2 and x_3 etc. we get

$$x = c_0 + \cfrac{1}{c_1 + \cfrac{1}{x_2}} = c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \cfrac{1}{x_3}}} \cdots = c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \cfrac{1}{c_3 + \cdots}}}. \tag{B2}$$

If x is a rational number (ratio of two integers) the continued fraction will eventually terminate. If x is irrational (like π) the continued fraction will go on for ever. The first few continued fraction coefficients $c_i (i = 0, 1, 2 \cdots)$ for π are

$$3, 7, 15, 1, \cdots . \tag{B3}$$

It is a property of continued fractions, which you can verify, that if a relatively large coefficient appears at some point, stopping the continued fraction at the previous coefficient gives an accurate approximation to the number. For, example, omitting 15 and subsequent coefficients in Eq. (B3) gives the result

$$\mathbf{3} + \frac{1}{\mathbf{7}} = \frac{22}{7}\,, \tag{B4}$$

(where the continued fraction coefficients are in bold) which is well known to be a fairly good approximation for π:

$$\pi = 3.14159\ldots, \qquad \frac{22}{7} = 3.14286\ldots. \tag{B5}$$

In the present case we are interested in the continued fraction representation of $y/2^n$, which is a rational fraction so the continued fraction will eventually terminate. However, the value of $y/2^n$ is close to m/r where r is no bigger than N (N can be represented by n_0 qubits with $n_0 = n/2$). So we are interested in a continued fraction *approximation* to $y/2^n$ with a denominator no bigger than N. (Recall that $2^n = (2^{n_0})^2$ which is greater than N^2.)

Consider the example described in this handout which has $N = 91, a = 4$ and $n = 14$ so $2^n = 16384$. The most probable results for y are those in the column labeled "nearest integer" in Table II. Suppose the measurement of y gives the nearest integer for $m = 5$, i.e. 13653. The exact continued fraction coefficients of 13653/16384 are

$$0, 1, 4, 1, 1364, 2\,. \tag{B6}$$

To have a rational approximation with a relatively small denominator we should omit the coefficients starting from the huge value of 1364. This gives

$$\frac{y_m}{2^n} \simeq \mathbf{0} + \cfrac{1}{\mathbf{1} + \cfrac{1}{\mathbf{4} + \cfrac{1}{\mathbf{1}}}} = \cfrac{1}{1 + \cfrac{1}{5}} = \frac{5}{6}\,, \tag{B7}$$

(continued fraction coefficients in bold) which tells us, if m and r have no common factors, that $m = 5$ and $r = 6$. We check if $r = 6$ works by directly calculating $4^6 (\mod 91)$. We find that it is equal to 1, see Eq. (27f), so the period is indeed 6. According to Appendix M in Mermin[2] *Quantum Computer Science* the probability of two large randomly chosen numbers not having a common factor is greater than 1/2. If we are unlucky and the assumption of no common factor does not work, then usually we would only have to try a few values for the common factor i.e. $2, 3, 4, \cdots$, before succeeding. If we are *really* unlucky, and the common factor is very large, we would give up at some point, start again and get a different value for y. In the related example studied in detail in Sec. VI, where the measurement is assumed to give the nearest integer to the second peak, the common factor turns out to be 2.

Appendix C: Unimportance of Small Phase Errors

The action of the controlled-phase gate is given by Eq. (11) and the QFT requires, in principle, these gates for $d = 1, 2, \cdots, n-1$. The total number of controlled phase gates is therefore $1 + 2 + \cdots + n - 1 = O(n^2)$. However, it is clearly impossible to accurately construct a phase gate for a phase which is exponentially small in n if n is large. For factoring, n would typically be several thousand.

Fortunately it is not necessary to include controlled phase gates with such small phase changes. Mermin [2] shows that one can generate the closest integer to a multiple of $2^n/r$ within almost the same probability as when one includes all gates (reduced by at most 1%) if one neglects controlled phase gates with $d > d^\star = \log_2(Cn)$, where the constant C is quite large (500π) but independent of n. Hence, in practice, one only needs of order $d^\star n$ controlled phase gates ($\sim n \log_2 n$) to obtain the desired result, rather than $O(n^2)$ which would be needed if one includes all the gates with d up to n. Hence the size of the circuit does not grow much faster than n which is a huge improvement compared with $O(n^2)$ if n is several thousand.

---

[1] P. W. Shor, in *Proc. 35th Symp. on Foundations of Computer Science*, edited by S. Goldwasser (IEEE Computer Society Press, Los Alamitos CA, 1994), p. 124.

[2] N. D. Mermin, *Quantum Computer Science* (Cambridge University Press, Cambridge, 2007).

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

[4] R. Vathsan, *Introduction to Quantum Physics and Information Processing* (CRC Press, Boca Raton, 2016).