Exact quantum Fourier transforms and discrete logarithm algorithms

Michele Mosca and Christof Zalka

Department of Combinatorics and Optimization
University of Waterloo, Waterloo, Ontario
Canada N2L 3G1
e-mail: mmosca@iqc.ca and zalka@iqc.ca

January 17, 2003

Abstract

We show how the quantum fast Fourier transform (QFFT) can be made exact for arbitrary orders (first for large primes). For most quantum algorithms only the quantum Fourier transform of order 2^n is needed, and this can be done exactly. Kitaev [9] showed how to approximate the Fourier transform for any order. Here we show how his construction can be made exact by using the technique known as "amplitude amplification". Although unlikely to be of any practical use, this construction e.g. allows to make Shor's discrete logarithm quantum algorithm exact. Thus we have the first example of an exact non black box fast quantum algorithm, thereby giving more evidence that "quantum" need not be probabilistic.

We also show that in a certain sense the family of circuits for the exact QFFT is uniform. Namely the parameters of the gates can be calculated efficiently.

1 Introduction

The "quantum fast Fourier transformation" (QFFT) plays an important role in quantum algorithms. It is a unitary transformation that applies the discrete Fourier transform to the amplitudes of a quantum register. The standard version has order 2^n and is applied to a quantum register consisting of n qubits. It was found by Coppersmith [5] (see also Shor [11]). The construction is essentially identical to the standard classical fast Fourier transform (FFT). Like the FFT it generalises to orders which are a power of a small prime and more generally to smooth numbers, thus integers who have only small prime factors (see Cleve [4]). These constructions implement the desired unitary transformation exactly.

In contrast, so far no exact (and efficient) constructions for arbitrary orders have been known. For his "Abelian stabiliser problem" Kitaev [9] gave an approximate implementation based on "eigenvalue estimation". Here we show how this eigenvalue estimation step can be made exact using "amplitude amplification". Amplitude amplification [1] is a slight generalisation of Grover's algorithm, allowing to apply the square root speed up to any heuristic algorithm. Brassard and Høyer [2] used a variant of it to make Simon's algorithm exact.

Finally we point out that an exact quantum Fourier transform for large prime orders can be used to make Shor's discrete logarithm algorithm exact.

2 The exact QFFT_p for large prime p

The quantum Fourier transform of order (or "modulus") N acts on "computational" basis states $|x\rangle$ as follows:

$$\mathrm{QFFT}_N: \qquad |x\rangle \quad \to \quad |\Psi_x\rangle = \frac{1}{N}\sum_{y=0}^{N-1} e^{2\pi i \frac{xy}{N}}|y\rangle.$$

For arbitrary, in particular non-smooth N, Kitaev [9] proposes to do this in two steps (second part of section 5 in [9], see also the review by Jozsa [8]):

$$|x\rangle \quad \to \quad |x, \Psi_x\rangle \quad \to \quad |\Psi_x\rangle$$

where, as usual, registers that "appear out of nowhere" are understood to have been initialised in the standard state $|0\rangle$. Similarly in the second step, one of the registers is reset to this state and can thus again be left away.

The first step constructs the Fourier state $|\Psi_x\rangle$ for a given x. This can be done exactly by first obtaining the "uniform amplitude" superposition $|\Psi_0\rangle$ of the first p basis states of a register and then "rephasing" it:

$$|x, 0\rangle \quad \to \quad |x, \Psi_0\rangle \quad \to \quad |x, \Psi_x\rangle. \tag{1}$$

As pointed out by Kitaev, $|\Psi_0\rangle$ can be obtained from $|0\rangle$ by a sequence of SO(2) rotations applied to each qubit in order from high to low significance, whereby the rotation angle has to be controlled by the previously touched qubits. The rephasing then simply consists of a rephasing on each qubit, proportional to x and the place value of the qubit.

The second step of Kitaev's construction is the reverse of

$$|\Psi_x, 0\rangle \quad \to \quad |\Psi_x, x\rangle.$$

This is done through a technique known as "eigenvalue estimation" (see also the article by Cleve et al. [3]), which details how to find the eigenvalue of an unknown eigenstate of some unitary U. We will describe this in more detail later. Here we only need to note that although this operation is not exact, it leaves the eigenstate $|\Psi_x\rangle$ unchanged. Thus it does:

$$|\Psi_x, 0\rangle \quad \to \quad |\Psi_x\rangle \sum_{x'} c_{x,x'}|x', g_{x,x'}\rangle \tag{2}$$

where on the right hand side the superposition should be dominated by the term with $x' = x$, such that a measurement would yield x with good probability. We also included some (unwanted) "garbage" $g_{x,x'}$ which may be produced along with the eigenvalue.

2.1 Using amplitude amplification

We now use "amplitude amplification" [1] to eliminate all but the desired term $|x, g_{x,x}\rangle$. We give here a quick review of this generalisation of Grover's algorithm. We are given a unitary operator A which, when applied to the initial state $|0\rangle$, gives an output state which has some component in a "good" subspace. Thus the probability $|P_{good}\ A|0\rangle|^2$ is not too small, where P_{good} is the projector onto the good subspace. The amplitude of the good component can be increased through the following procedure

$$\left[\ A(\mathbf{1} + (e^{i\phi} - 1)|0\rangle\langle 0|)A^{-1}\ (\mathbf{1} + (e^{i\varphi} - 1)P_{good})\ \right]^T A|0\rangle$$

where the sequence of operations in the brackets is repeated T times, depending on the "success probability" of the "algorithm" A alone. As in Grover's algorithm, the fastest increase is achieved when both phases are chosen $\phi = \varphi = \pi$. The algorithm can be analysed by noting that the state always remains in a subspace spanned by the state we are seeking $P_{good}\ A|0\rangle$ and by $(1 - P_{good})\ A|0\rangle$. Usually an integer number of iterations will not lead exactly to the desired state and so we need to chose different (non-optimal) phases, either in all steps or only in the last one or two. In our case we will leave the phases at their standard settings, but will modify A so that its success probability is reduced to 1/4 where a single iteration leads exactly to the desired state.

The operator A will be given by eq. 2, where the state $|\Psi_x\rangle$ will have to be added as a "spectator" that is not changed.

2.1.1 "Recognising" the correct solution

Apart from the "heuristic" algorithm A, amplitude amplification requires a way to "recognise" the good states. More precisely, we need a way to apply the phase $e^{i\varphi}$ to the good subspace and leave its orthogonal complement unchanged. So how can we check whether a number x' is the right eigenvalue of $|\Psi_x\rangle$, thus whether $x' = x$? This can be done because the eigenstate $|\Psi_x\rangle$ is still available exactly. Thus given a state of the form $|\Psi_x\rangle \sum_{x'} c_{x,x'} |x', g_{x,x'}\rangle$, we can check the second register against the first one. To do this we apply the reverse of the steps in eq. 1 to these two registers, thus:

$$|x', \Psi_x\rangle \quad \rightarrow \quad |x', \Psi_{x-x'}\rangle \quad \rightarrow \quad |x', \theta_{x-x'}\rangle$$

where in the second step we only act on the second register. The state $|\Psi_0\rangle$ is mapped back to $|0\rangle$, while for $x' \neq x$ we get some state $|\theta_{x-x'}\rangle$ orthogonal to $|0\rangle$. We can now apply the phase $e^{i\varphi}$ to the $|0\rangle$ state and undo the previous operations.

2.2 "Uniformising" the success probability

One obstacle to using amplitude amplification to make algorithms exact is that the success probability of the "heuristic" algorithm A must be known. But this probability may depend on the (unknown) instance of the problem. In our case the success probability of eigenvalue estimation on $|\Psi_x\rangle$ indeed does depend on x. We can fix this problem by modifying A such that the new success probability will become instance independent and equal to the average over all instances for the original A. To do this uniformisation we pick an integer r uniformly at random from $\{0, 1, \dots p-1\}$ and replace $|\Psi_x\rangle$ with $|\Psi_{x+r}\rangle$, which is just a rephasing. We keep a record of r and subtract it again from the result of eigenvalue estimation. To do this with a unitary A we will need an additional register for r, but this is no problem, as we have already included the possibility that eigenvalue estimation (eq. 2) also generates some unwanted garbage $g_{x,x'}$.

So now exact amplitude amplification will allow us to do

$$|\Psi_x, 0\rangle \quad \rightarrow \quad |\Psi_x\rangle|x, g_{x,x}\rangle.$$

To get rid of the "garbage" we can do the usual trick of copying the wanted result x into an additional "save" register and then undoing the previous steps. In total this will lead to six applications of A for an exact QFFT.

In summary, the construction of an exact QFFT relies on making eigenvalue estimation (on Fourier states $|\Psi_x\rangle$) exact. The essential observations are that eigenvalue estimation leaves the eigenstate $|\Psi_x\rangle$ exactly unchanged and so it can be used for the checking stage of amplitude amplification. Furthermore we used that the success probability of estimating x from $|\Psi_x\rangle$ can rather easily be "uniformised" across all $x = 0 \dots p-1$.

3 An exact discrete logarithm algorithm

An exact algorithm for the QFFT leads in a straightforward manner to an exact algorithm for the discrete logarithm algorithm of the same order. This was also observed for finite fields of prime order by Brassard and Høyer [2] (Theorem 12). For smooth orders (only small prime factors) the problem can easily be solved classically. Here we give a quick review for the case when the order is a large prime (see also [12], section 2.2.3).

In a discrete logarithm problem we are given an element α which generates a cyclic group of some finite order, here a prime. Thus $\alpha^p = e$. Then another element β of the group is given and we want to know which power of α it is; that is, the integer a for which $\beta = \alpha^a$. This is also written as $a = \log_\alpha \beta$. In the quantum solution (see Shor [11]), we prepare two registers, each in a uniform amplitude superposition of p basis states:

$$\frac{1}{p}\sum_{x=0}^{p-1}\sum_{y=0}^{p-1}|x, y\rangle.$$

Then we compute the function $\alpha^x \beta^y$ in an additional register and measure it. This will leave the two registers in a superposition of the form $\sum_y |x_0 - a \cdot y, y\rangle$ where all arithmetic operations are understood to be modulo p, x_0 is random and y runs over $0 \dots p-1$. By Fourier transforming each register with a QFFT_p we get a similar state but without the offset x_0, namely an equally weighted superposition of all states of the form $|x, a \cdot x\rangle$ with $x = 0 \dots p-1$. A measurement will now allow to compute a in all cases except when $x = 0$. Thus we have the known and instance independent success probability of $1 - 1/p$, which allows to easily make the algorithm exact by using (exact) amplitude amplification.

3.1 Alternatively: directly uniformising dlog

Actually one can directly make the success probability of the dlog algorithm instance independent. Thus one uses the usual algorithm with a QFFT_{2^n}, but replaces β with $\beta \cdot \alpha^r$ where r is again chosen uniformly at random from $0 \dots p-1$. We have noted this approach a while ago, but were not able to show that the (now averaged) success probability can be computed efficiently, thus it is not clear whether the circuit for a given p can be computed efficiently.

4 Eigenvalue estimation

In our case we want to estimate the eigenvalue of $|\Psi_x\rangle$ under the (unitary) cyclic shift operator U which acts on computational basis states as: $|x\rangle \to |(x+1) \bmod p\rangle$. For eigenvalue estimation we need to do large powers of U, which in this case is easy. Namely we first prepare an auxiliary n-qubit register in a uniform amplitude superposition of all its $N = 2^n$ basis states. (We will choose N to be larger than p, see below.) Then we do:

$$\frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} |y\rangle \; |\Psi_x\rangle \quad \to \quad \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} |y\rangle \; U^y |\Psi_x\rangle \quad = \quad \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{-2\pi i \frac{xy}{p}} |y\rangle \; |\Psi_x\rangle.$$

where we used that the eigenvalue of $|\Psi_x\rangle$ under U is $e^{-2\pi i x/p}$. Note that the operation we have to do is simply a modular addition on computational basis states, thus $|a, b\rangle \to |a, (a+b) \bmod p\rangle$. After a Fourier transform of size 2^n on the auxiliary register, the probability of measuring y would be given by:

$$p_y = f^2(y - xN/p) \qquad \text{where} \quad f(z) = \frac{\sin(\pi z)}{N \sin(\pi z/N)}.$$

We illustrate the function $f(z)$ with $N \to \infty$ in figure 1. It is peaked around $z = 0$ so that after measuring some y we would guess for the number we want to find $x \approx y \cdot p/N$. The choice with the highest probability of obtaining the correct x would be to simply round $y \cdot p/N$ to the closest integer. Partially to simplify notation, here we round up to the next integer, thus our guess is $x' = \lceil y \cdot p/N \rceil$. (For us the loss of some success probability does not matter, at least not as long as it is at least $1/4$.)

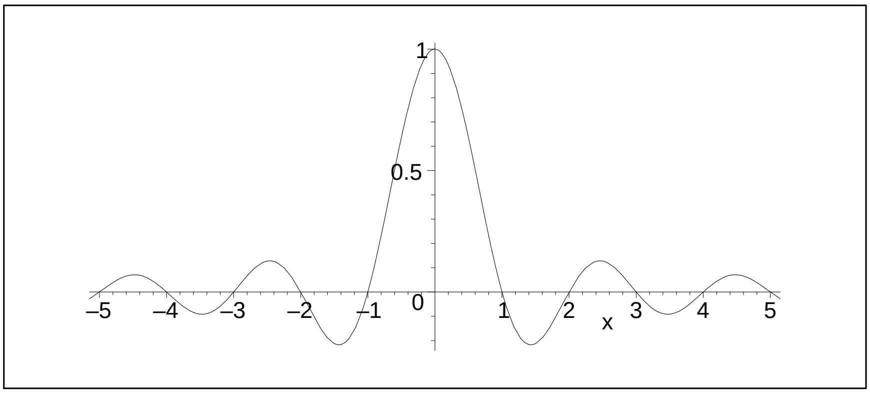

Figure 1: The function $\frac{\sin(\pi x)}{\pi x}$.

Because we have $N > p$, it is clear that if we measure $y = \lfloor x \cdot N/p \rfloor$ we will calculate the correct x. For a given x, smaller y may also lead to the correct x, but here we would like to eliminate this contribution to the success probability, as it will lead to a simpler expression. Given a y it is possible to eliminate these cases by also checking $y > \lceil y \cdot p/N \rceil \cdot N/p - 1$ and "throwing away" y's which do not satisfy this. (Note that in order to obtain an algorithm A with a certain success probability, we can think as if this were a non-reversible algorithm including measurements and classical computations. Such an algorithm can then easily be turned into a unitary A which, besides the intended answer, also produces some "garbage".) So now the success probability p_x for correctly getting x from $|\Psi_x\rangle$ is:

$$p_x = f^2(\lfloor xN/p \rfloor - xN/p) = f^2(\frac{xN \bmod p}{p}).$$

To get the instance independent success probability of the uniformised algorithm, we average this over all $x = 0 \ldots p-1$:

$$\bar{p} = \frac{1}{p} \sum_{x=0}^{p-1} f^2(\frac{xN \bmod p}{p}) = \frac{1}{p} \sum_{k=0}^{p-1} f^2(k/p)$$

where we have used that N and p are coprime and so for each x there is exactly one k.

4.1 Efficiently calculating the success probability

For large p this sum of course is well approximated by the corresponding integral, which (for large N) is approximately 0.4514. Here we show that for each p and N, the success probability can be approximated efficiently in the sense that the computation time is polynomial in the number of (e.g. decimal) digits we want to compute. The following method achieves this in a simple way, although it is probably not the best one could do. Note that $f^2(z)$ can be expanded in a (fast converging) power series in x and $1/N$. (To compute $\bar{p}$ to d digits we will only use polynomially many terms in d.) Now each power z^m of z can be summed separately, giving:

$$\frac{1}{p}\sum_{k=0}^{p-1}(k/p)^m \;=\; \frac{1}{p^{m+1}}\; S_m(p) \;=\; \frac{1}{p^{m+1}}\sum_{i=0}^{m+1} A_{m,i}\; p^i$$

where for each power m the coefficients $A_{m,i}$ can be calculated (in various ways) in time polynomial in m. (A straightforward way is to simply solve the equations resulting from $S_m(p+1) - S_m(p) = p^m$ for the $A_{m,i}$. E.g. for $m = 1$ we get the familiar formula $\sum_{k=0}^{p-1} k = p(p-1)/2$.)

4.2 Adjusting the success probability

Once we have calculated the success probability $\overline{p}$ (to arbitrary precision) for a given p, we can use this to modify the algorithm A so that it will succeed exactly with probability $1/4$, so that just one iteration of amplitude amplification leads to an exact algorithm. One way to do this is to add a qubit prepared in state $\cos(\alpha)|0\rangle + \sin(\alpha)|1\rangle$ with $\overline{p}\sin^2(\alpha) = \frac{1}{4}$ and additionally require for success that this qubit be in state $|1\rangle$. The preparation of this qubit will now require the one "strange" gate in our algorithm, although its rotation angle α can be computed efficiently in the above sense.

5 Further remarks and observations

5.1 Generalisation to arbitrary orders

The construction of the exact QFFT_q easily generalises to arbitrary orders q. Above we only needed the primality of the order for (efficiently) computing the success probability. And there we only needed that $N = 2^n$ and q should be coprime. Things can easily be adjusted for the case when q is even. Either we can modify (a bit) the calculation of the success probability, or we can consider the QFFT_q as a tensor product of a QFFT with odd order and a standard one with order a power of 2. Similarly, of course, we can generalise to QFFT's over finite Abelian groups, not just cyclic ones.

Also the discrete logarithm algorithm can be generalised to arbitrary orders q. Given the exact QFFT_q, the algorithm will be successful whenever the first number in the measured pair $(x, xa \bmod q)$ is coprime to q. So the success probability is $\phi(q)/q$ where $\phi(q)$ is the Euler totient function. If we know the factorisation of q, this is easily calculated and so amplitude amplification can be used to make the algorithm exact.

5.1.1 Factorisation of the order of the dlog not known

In the following we give a more involved solution for the case when the factorisation of the order q is not known. It consists of $O(\log q)$ runs of (variants of) the dlog quantum circuit. What is important is, that these variants still only use the special gates calculated (efficiently) at the beginning from q.

In the first run it is enough, as before, to use amplitude amplification only to get rid of the case $x = 0$. We now measure a pair $(x, xa \bmod q)$. If x is coprime to q we can directly calculate a and are done. If $\gcd(x, q) = d > 1$, we still get some information about a, namely $a' = a \bmod q/d$, and of course the factor d of q. Now we have $a = a' + a'' \cdot q/d$, where, in a standard way, a'' can be found by solving the dlog problem with $\tilde{\alpha} = \alpha^{q/d}$ and $\tilde{\beta} = \beta\alpha^{-a'}$. This dlog problem has smaller order, as $\tilde{\alpha}^d = e$, but we want to reuse the original quantum circuit for order q. If in this original circuit we simply replace α, β with $\tilde{\alpha}, \tilde{\beta}$, we get (after the two QFFT's):

$$\frac{1}{\sqrt{d}} \sum_{k=0}^{d-1} |k \cdot q/d,\ a''k \cdot q/d \mod q\rangle$$

(Note that this is essentially the same as $\sum_k |k, a''k \bmod d\rangle$.) We want to avoid only the case $k = 0$, but in order not to introduce new "special" gates, we prefer to eliminate 3/4 of all states, such that one step of standard amplitude amplification will lead to an exact solution. We can e.g. only retain the last quarter of the values $k = 0 \dots d-1$, although, if d is not divisible by 4, we will have to "partially tag" some states. (This can be done by appending a qubit in state $c|0\rangle + s|1\rangle$ with $|s|^2 = 1/4, 1/2$ or $3/4$.)

Now, like in the first step, we will either directly get a'', or will gain partial information on a'', together with a factor of d. This can be iterated (at most $O(\log q)$ times) till the order of the dlog problem is small.

Note that in our construction we have taken care not to introduce new "special" gates during the computation. This means that really the $O(\log q)$ quantum runs can be put together into one quantum circuit whose gates can be computed from q alone (without knowing its factorisation).

5.2 No exact factorisation algorithm

Let us also note that it is not clear how to make Shor's integer factorisation algorithm exact with the techniques used here. Thus this is a challenge that remains. We note that Mosca [10] shows how to make factorisation exact in a slightly generalised model of exact quantum computation.

5.3 Review of other work on quantum Fourier transforms

It is interesting to note that after Kitaev [9] a more efficient and probably also more natural way to approximate the QFFT for arbitrary orders has been given by Hallgren and Hales [7]. In particular their construction uses fewer qubits, but it seems not to lend itself to the techniques used here to make it exact.

Also note the simplified "semiclassical" version of the standard QFFT by Griffiths and Niu [6]. For practical implementations of Shor's algorithms this would probably be the method of choice.

Acknowledgements

The two authors have independently found the results described here, although Ch.Z. acknowledges inspiration by related work of M.M. on using amplitude amplification to make algorithms exact. Ch.Z. would like to thank Professor David Jackson of our department for discussions on summing the m$^{\mathtt{th}}$ powers of the first n integers. He is supported by CSE (Communications Security Establishment) and MITACS (Mathematics of Information Technology and Complex Systems), both from Canada. M.M. thanks Richard Cleve, Lisa Hales, and John Watrous for discussions at MSRI that prompted him to think about solving the QFFT exactly in this context. M.M. is the Canada Research Chair in Quantum Computation and is supported by NSERC, MITACS, CFI, ORDCF, and PREA.

References

[1] G. Brassard, P. Høyer and A. Tapp, *Quantum Counting*, ICALP'98, (also quant-ph/9805082)

[2] G. Brassard and P. Høyer, *An Exact Quantum Polynomial-Time Algorithm for Simon's Problem*, ISTCS'97, (also quant-ph/9704027)

[3] R. Cleve, A. Ekert, C. Macchiavello and M. Mosca, *Quantum Algorithms revisited*, Proc. R. Soc. Lond. A (1998) **454**, pp. 339-354, (also quant-ph/9708016)

[4] R. Cleve, *A note on computing Fourier transforms by quantum programs* Unpublished (1994)
(available at `http://pages.cpsc.ucalgary.ca/~cleve/papers.html`)

[5] D. Coppersmith, IBM Research Report RC 19642 (1994) (also quant-ph/0201067)

[6] R. B. Griffiths and C. Niu, *Semiclassical Fourier Transform for Quantum Computation*, Phys. Rev. Lett. **76** (1996) pp.3228-3231 (also quant-ph/9511007)

[7] S. Hallgren and L. Hales, *An Improved Quantum Fourier Transform Algorithm and Applications*, FOCS 2000,
(also available at `http://www.cs.caltech.edu/~hallgren/`)

[8] R. Jozsa, *Quantum Algorithms and the Fourier Transform*, Proc. R. Soc. Lond. A (1998) **454**, pp. 323-337, (also quant-ph/9707033)

[9] A. Yu. Kitaev, *Quantum measurements and the Abelian Stabilizer Problem*, quant-ph/9511026

[10] M. Mosca, *On the Quantum Derandomization of Algorithms*, manuscript in preparation; based on presentation at MSRI workshop on Quantum Information Processing, Dec. 2002.

[11] P. Shor, *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*, Proc. 35th Annual Symposium on Foundations of Computer Science. IEEE Press, pp 124-134, Nov. 1994, (also quant-ph/9508027)

[12] Ch. Zalka and J. Proos, *Shor's discrete logarithm quantum algorithm for elliptic curves*, manuscript in preparation