QUANTUM INFORMATION & COMPUTATION

Nilanjana Datta, DAMTP Cambridge

1 The quantum Fourier transform and periodicities

1.1 Quantum Fourier transform mod N

The quantum Fourier transform (QFT) can be viewed as a generalisation of the Hadamard operation to dimensions $N > 2$. Later we will be especially interested in $N = 2^n$ i.e. the QFT on an n-qubit space. As a pure mathematical construction it is the same as the so-called discrete Fourier transform which is widely used in digital signal and image processing. It is a *unitary* matrix that arises naturally in a wide variety of mathematical situations so it fits well into the quantum formalism, providing a bridge between a quantum operation and certain mathematical problems. In fact QFT is at the heart of most known quantum algorithms that provide a significant speedup over classical computation.

Let $\mathcal{V}_N$ denote a state space with an orthonormal basis $\{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$ labelled by $\mathbb{Z}_N$. The quantum Fourier transform (QFT) modulo N, denoted QFT_N (or just QFT when N is clear) is the unitary transform on $\mathcal{H}_N$ defined by:

$$QFT : |x\rangle \to \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \exp(2\pi i \frac{xy}{N})\ |y\rangle \tag{1}$$

Thus the jk^{th} matrix entry is

$$[\text{QFT}]_{jk} = \frac{1}{\sqrt{N}} \exp\left(2\pi i \frac{jk}{N}\right) \qquad j, k = 0, \ldots, N-1$$

(note: here we are labelling rows and columns from 0 to $N-1$ in $\mathbb{Z}_N$ rather than 1 to N.) If $\omega = e^{2\pi i/N}$ is the primitive N^{th} root of unity then the matrix elements are all powers of ω (divided by $\sqrt{N}$) following a simple pattern:

- The initial row and column always contain only 1's.
- Each row (or column) is a geometric sequence. The k^{th} row (or column) for $k = 0, \ldots, N-1$ is the sequence of powers of ω^k (starting with power 0 up to power $N-1$).

Many properties of QFT, including the fact that it is unitary, follow from a basic algebraic fact about roots of unity and geometric series. Recall the formula for the sum of any geometric series

$$1 + \alpha + \alpha^2 + \ldots + \alpha^{N-1} = \begin{cases} \frac{1-\alpha^N}{1-\alpha} & \text{if } \alpha \neq 1 \\ N & \text{if } \alpha = 1 \end{cases}$$

Then setting $\alpha = w^K$ (for some chosen K) we have $\alpha = 1$ iff K is a multiple of N. Thus

$$1 + \omega^K + \omega^{2K} + \ldots + \omega^{(N-1)K} = \begin{cases} N & \text{if } K \text{ is a multiple of } N. \\ 0 & \text{if } K \text{ is not a multiple of } N. \end{cases} \tag{2}$$

Now to see that QFT is unitary, consider the jk^{th} element of the matrix product QFT†QFT. This is $1/N$ times the sum of "the j^{th} row of QFT† lined up against the k^{th} column of QFT". The latter sum is just the geometric series with $\alpha = \omega^{k-j}$, divided by N. So using eq. (2) we get $0/N = 0$ if $k \neq j$ and we get $N/N = 1$ if $k = j$ i.e. QFT†QFT is the identity matrix and QFT is unitary.

1.2 Periodicity determination

A fundamental application of the Fourier transform (both classically and quantumly) is the determination of periodicity exhibited in a function or some other given data. Some important mathematical problems (such as integer factorisation, as we will see later) can be reduced to problems of periodicity determination.

Suppose we are given (a black box for) a function $f : \mathbb{Z}_N \to Y$ (where typically $Y = \mathbb{Z}_M$ for some M) and it is promised that f is periodic with some period r i.e. there is a smallest number r such that $f(x + r) = f(x)$ for all $X \in \mathbb{Z}_N$ (and $+$ is addition mod N). We will also assume that f is one-to-one in each period i.e. $f(x_1) \neq f(x_2)$ for all $0 \leq x_1 < x_2 < r$. We want a method of determining r with some constant level of probability (0.99 say) that is independent of increasing the size of N. It can be shown that $O(\sqrt{N})$ queries to f (i.e. a number not bounded by any polynomial in $\log N$)) are necessary and sufficient to achieve this in *classical* computation with a black box for f. In some cases further information may be available about f e.g. we may have an explicit formula for it but the periodicity determination may *still* be hard (we will see an example later), requiring a number of steps that is not bounded by any polynomial in $\log N$. In the quantum scenario we will see that r can always be determined with any constant high level of probability $1 - \epsilon$ using only $O(\log \log N)$ queries and poly($\log N$) further processing steps i.e. exponentially faster than any classical method.

Quantum algorithm for periodicity determination

We begin by constructing a uniform superposition $\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$ and one query to U_f to obtain the state $|f\rangle = \frac{1}{\sqrt{N}} \sum_{\text{all}x} |x\rangle |f(x)\rangle$. Since f is periodic (with unknown period r) r must divide N exactly and we set $A = N/r$, which is the number of periods. If we measure the second register we will see some value $y = f(x_0)$ where x_0 is the least x having $f(x) = y$. Then the first register will be projected into an equal superposition of the A values of $x = x_0, x_0 + r, x_0 + 2r, \ldots, x_0 + (A-1)r$ for which $f(x) = y$ i.e. we get

$$|per\rangle = \frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |x_0 + jr\rangle$$

Here $0 \leq x_0 \leq r - 1$ has been chosen uniformly at random (by the extended Born rule, since each possible value y of f occurs the same number A of times i.e. once in each period.) If we measure the register of $|per\rangle$ we will see $x_0 + j_0 r$ where j_0 has been picked uniformly at random too. Thus we have a random period (the j_0^{th} period) and a random element in it (determined by x_0) i.e. overall we get a random number between 0 and

$N-1$, giving no information about r at all. Nevertheless the state $|per\rangle$ seems to contain the information of r!

The resolution of this problem is to use the Fourier transform which is known even in *classical* image processing, to be able to pick up periodicities in a periodic pattern irrespective of an overall random shift of the pattern (e.g. the x_0 in $|per\rangle$). Applying QFT to $|per\rangle$ we get (using eq. (1) with x replaced by $x_0 + jr$, and summing over j):

$$QFT\,|per\rangle = \frac{1}{\sqrt{NA}} \sum_{j=0}^{A-1} \left(\sum_{y=0}^{N-1} \omega^{(x_0+jr)y} \,|y\rangle \right) = \frac{1}{\sqrt{NA}} \sum_{y=0}^{N-1} \omega^{x_0 y} \left[\sum_{j=0}^{A-1} \omega^{jry} \right] |y\rangle\,. \qquad (3)$$

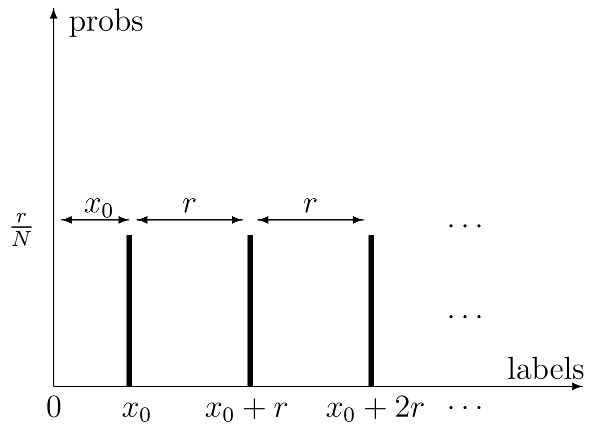
(In the last equality we have reversed the order of summation and factored out the j-independent $\omega^{x_o y}$ terms). Which labels y appear here with nonzero amplitude? Look at the square-bracketed coefficient of $|y\rangle$ in eq. (3). It is a geometric series with powers of $\alpha = e^{2\pi i r y/N} = (e^{2\pi i/A})^y$ summed from power 0 to power $A-1$. According to eq. (2) (now applied with A taking the role of N there) this sum is *zero* whenever y is not a multiple of A and the sum is A otherwise i.e. only multiples of $A = N/r$ survive as y values:

$$\sum_{j=0}^{A-1} \omega^{jry} = \begin{cases} A & \text{if } y = kN/r \text{ for } k = 0, \ldots, r-1 \\ 0 & \text{otherwise} \end{cases}$$

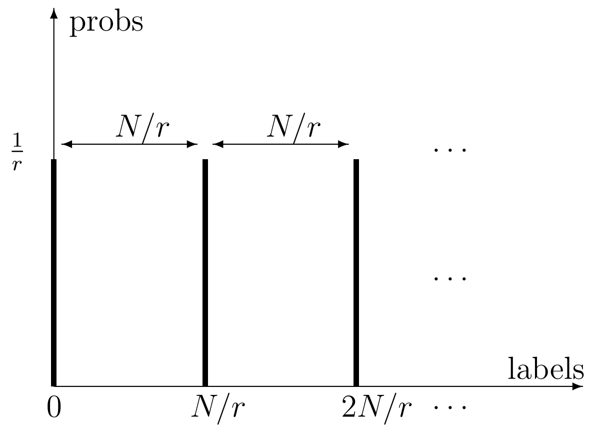
and

$$\text{QFT}\,|per\rangle = \sqrt{\frac{A}{N}} \sum_{k=0}^{r-1} \omega^{x_0(kN/r)} \,|kN/r\rangle\,.$$

The random shift x_0 has been eliminated from the labels and now occurs only in a pure phase $\omega^{x_0 kN/r}$ (whose modulus squared is 1), and the periodicity of the ket labels has been "inverted" from r to $A = N/r$. Since measurement probabilities are squared moduli of the amplitudes, these probabilities are now independent of x_0 and depend only on N (known) and r (to be determined). This is represented schematically in the following diagram.



(a) for $|per\rangle$



(b) for QFT$|per\rangle$

If we now measure the label we will obtain a value c which is a multiple k_0N/r of N/r where $0 \leq k_0 \leq r-1$ has been chosen uniformly at random. Thus $c = k_0N/r$ so

$$\frac{k_0}{r} = \frac{c}{N}.$$

Here c and N are known and k_0 is unknown and random, so how do we get r out of this? If (by some good fortune!) k_0 was coprime to r we could cancel c/N down and read off r as the denominator. If k_0 is not coprime to r then this procedure will deliver a denominator r' that is smaller than the correct r so $f(x) \neq f(x+r')$ for any x. Thus in our process we check the output r value by evaluating $f(0)$ and $f(r)$ and accepting r as the correct period iff these are equal.

But k_0 was chosen at random so what is the chance of getting this good fortune of coprimality? We'll use (without proof) the following theorem from number theory:

Theorem 1 *(Coprimality theorem) The number of integers less than r that are coprime to r grows as $O(r/\log\log r)$ with increasing r. Hence if $k_0 < r$ is chosen at random*

$$\textit{prob}(k_0 \textit{ coprime to } r) \ \approx O((r/\log\log r)/r) = O(1/\log\log r). \qquad \square$$

Thus if we repeat the whole process $O(\log\log r) < O(\log\log N)$ times we will obtain a coprime k_0 in at least one case with a constant level of probability. Here we have used the following fact from probability theory:

Lemma 1 *If a single trial has success probability p and we repeat the trial M times independently then for any constant $0 < 1-\epsilon < 1$:*

$$\textit{prob(at least one success in } M \textit{ trials)} > 1-\epsilon \textit{ if } M = \frac{-\log\epsilon}{p}$$

so to achieve any constant level $1-\epsilon$ of success probability, $O(1/p)$ trials suffice.

Proof of lemma We have that the probability of at least one success in M runs $= 1-$ prob(all runs fail) $= 1-(1-p)^M$. Then $1-(1-p)^M = 1-\epsilon$ if $M = \frac{-\log\epsilon}{-\log(1-p)}$. Hence any number of trials $\geq \frac{-\log\epsilon}{-\log(1-p)}$ suffices. Now all we want to do is to get a simple estimate of this threshold value $\frac{-\log\epsilon}{-\log(1-p)}$. To do this we use the fact that $p < -\log(1-p)$ for all $0 < p < 1$ to see that

$$\frac{-\log\epsilon}{-\log(1-p)} < \frac{-\log\epsilon}{p}.$$

This allows us to infer that $M = O(1/p)$ repetitions suffice. $\square$

In each round we query f three times (once at the start to make $|f\rangle$ and twice more at the end to check the output r) so we use $O(\log\log N)$ queries in all. We also need to apply the "large" unitary gate QFT_N (which grows with N) and we show in the next section that this may be implemented in $O((\log N)^2)$ elementary steps. The remaining operations are all familiar arithmetic operations on integers of size $O(N)$ (such as cancelling c/N down to lowest form) that are all well known to be computable in polynomial time i.e. $\mathrm{poly}(\log N)$ steps. Thus we succeed in determining the period with any constant level $1-\epsilon$ of success probability with $O(\log\log N)$ queries and $O(\mathrm{poly}(\log N))$ further computational steps.

We have described above the quantum algorithm for periodicity determination, for periodic functions on $\mathbb{Z}_N$ which will form the core of Shor's efficient quantum algorithm for integer factorisation (cf below). But the basic problem of periodicity determination may be mathematically generalised in a natural way from $\mathbb{Z}_N$ to an arbitrary group G as the so-called *hidden subgroup problem* (beyond the scope of this course). This formalism leads to a class of further important quantum algorithms of which Simon's algorithm and the above $\mathbb{Z}_N$ case are special cases.

1.3 Efficient implementation of QFT

This subsection is not required for exam purposes.

If $N = 2^n$ is an integer power of 2 then QFT mod N acts on n qubits. For these dimension sizes we will show how to implement QFT with a circuit of polynomial size $O(n^2)$. This is a very special property of QFT – almost all unitary transforms in dimension 2^n require exponential sized ($O(\mathrm{poly}(2^n))$ sized) circuits for their implementation. For general N (not a power of 2) we do not have an exact efficient (i.e. $\mathrm{poly}(\log N)$ sized) implementation. Instead we generally approximate QFT mod N by QFT mod 2^k where 2^k is near enough to N to incur only an acceptably small reduction in the success probability of the algorithm.

Our efficient implementation of QFT is really just a translation of the classical fast Fourier transform formalism to the quantum scenario. We begin by showing that the n qubit state

$$QFT\,|x\rangle = \frac{1}{\sqrt{2^n}}\sum_y \exp 2\pi i\frac{xy}{2^n}\;|y\rangle$$

is actually a *product* state of n one-qubit states. We write $0 \le x, y \le 2^{n-1}$ in binary (as n bit strings of digits):
(Warning: Take care to distinguish arithmetic mod 2^n in $\mathbb{Z}_{2^n}$ used here from the bitwise arithmetic of n bit strings that we used earlier!)

$$x = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \ldots + x_1 2 + x_0$$

$$y = y_{n-1}2^{n-1} + y_{n-2}2^{n-2} + \ldots + y_1 2 + y_0$$

In $xy/2^n$ we discard any terms that are whole numbers since these make no contribution to $\exp 2\pi i xy/2^n$ and a direct calculation gives:

$$\frac{xy}{2^n} \equiv y_{n-1}(.x_0) + y_{n-2}(.x_1x_0) + \ldots + y_0(.x_{n-1}x_{n-2}\ldots x_0) \qquad (4)$$

where the factors in parentheses are binary expansions e.g.

$$.x_2x_1x_0 = \frac{x_2}{2} + \frac{x_1}{2^2} + \frac{x_0}{2^3}$$

Now

$$\sum_y \exp 2\pi i \frac{xy}{2^n} \, |y\rangle = \sum_{y_0,\ldots,y_{n-1}} \exp 2\pi i \frac{xy}{2^n} \, |y_{n-1}\rangle \, |y_{n-2}\rangle \ldots |y_0\rangle$$

and we want to insert the expression for $xy/2^n$ from eq. (4) into the exponential. Since eq. (4) is a *sum* over the different y_i's, the exponential will be a *product* of these terms and hence the sum $\sum_{y_0,\ldots,y_{n-1}}$ splits up into a product of single index sums $(\sum_{y_0})(\sum_{y_1})\ldots(\sum_{y_{n-1}})$ so we get

$$\sum_y \exp 2\pi i \frac{xy}{2^n} \, |y\rangle = \sum_y \exp 2\pi i \frac{xy}{2^n} \, |y_{n-1}\rangle \, |y_{n-2}\rangle \ldots |y_0\rangle =$$

$$\left(|0\rangle + e^{2\pi i(.x_0)} \, |1\rangle\right) \left(|0\rangle + e^{2\pi i(.x_1x_0)} \, |1\rangle\right) \ldots \left(|0\rangle + e^{2\pi i(.x_{n-1}\ldots x_0)} \, |1\rangle\right). \quad (5)$$

Hence QFT$|x\rangle$ is the product of corresponding 1-qubit states obtained by taking each bracket with a $1/\sqrt{2}$ normalising factor.

This factorisation is the key to building our QFT circuit. It should map each basis (product) state $|x_{n-1}\rangle \ldots |x_0\rangle$ into the corresponding product state given in eq. (5). Before we start note that the Hadamard operation can be expressed in our binary fractional notation as

$$H \, |x\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i(.x)} \, |1\rangle\right).$$

Indeed if $x = 0$ resp. 1 then $.x$ is 0 resp. 1/2 as a decimal fraction so $e^{2\pi i(.x)}$ is 1 resp. -1, as required.

To see how the QFT circuit actually works, let's look at the example of $N = 8$ i.e. $n = 3$. We want a circuit that transforms $|x_2\rangle \, |x_1\rangle \, |x_0\rangle$ to the following states in these three registers (called y_2, y_1, y_0 at the output):

$$\underbrace{\overset{y_2 \text{ register}}{\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(.x_0)}|1\rangle\right)}}_{\text{STAGE 3}} \otimes \underbrace{\overset{y_1 \text{ register}}{\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(.x_1x_0)}|1\rangle\right)}}_{\text{STAGE 2}} \otimes \underbrace{\overset{y_0 \text{ register}}{\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(.x_2x_1x_0)}|1\rangle\right)}}_{\text{STAGE 1}}$$

$H|x_0\rangle$.
This operation depends
only on x_0 (not x_1, x_2).
Do it last (third) and
put result on x_0 line.

$H|x_1\rangle$ followed by
phase shift $e^{2\pi i 0.0x_0}$
i.e. phase shift of $e^{2\pi i 0.01}$
controlled by x_0 value.
These operations
depend on x_1, x_0 (not x_2).
Do them second and
accumulate result on x_1 line
(as x_1 line no longer
needed after this).

$H|x_2\rangle$ followed by
phase shifts of
$e^{2\pi i 0.01}$ and $e^{2\pi i 0.001}$
controlled by x_1 and x_0
respectively.
These operations
depend on x_0, x_1, x_2.
Do them first and
accumulate result on x_2 line
(as x_2 line no longer
needed after this).

After completion of these three stages, the desired final contents of the y_0, y_1, y_2 lines are respectively on the x_2, x_1, x_0 lines. Thus finally just reverse the order of the qubits in the string (e.g. by swap operations).

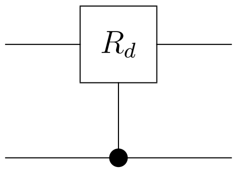
To draw an actual circuit diagram we consider the three stages in turn. In addition to the Hadamard gate H we'll introduce the 1-qubit phase gate:

$$R_d = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2^d} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i(0.00\ldots01)} \end{pmatrix} \qquad (6)$$

where the binary digit 1 in the last exponential is $(d+1)$ places to the right of the dot. The controlled-R_d gate, denoted C-R_d acts on two qubits and is defined by the following actions

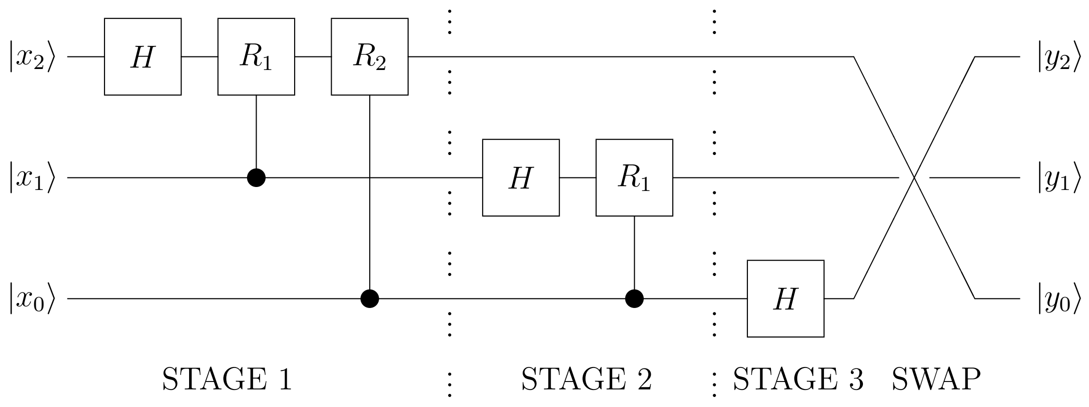
$$\text{C-}R_d\,|0\rangle\,|\psi\rangle = |0\rangle\,|\psi\rangle \qquad \text{C-}R_d\,|1\rangle\,|\psi\rangle = |1\rangle\,R_d\,|\psi\rangle$$

for any 1-qubit state $|\psi\rangle$. Diagramatically this will be denoted as



with a "blob" on the control qubit line.

In terms of all these, the circuit for QFT$_8$ is



For $N = 8 = 2^3$ we use 3 Hadamard gates (one in each stage) and $2+1$ controlled phase gates (in stages 1 and 2 respectively). For general $N = 2^n$ we would use n Hadamard gates (one in each of n stages) and $(n-1)+(n-2)+\ldots+2+1 = n(n-1)/2$ controlled phase gates (in stages $1, 2, \ldots, n-1$ respectively). Overall we have $O(n^2) = O((\log N)^2)$ gates for QFT mod N. (In this accounting we have ignored the final swap operation to reverse the order of qubits, but this requires only a further $O(n)$ 2-qubit SWAP gates to implement).