Quantum algorithms (CO 781/CS 867/QIC 823, Winter 2013)
Andrew Childs, University of Waterloo

LECTURE 2: The abelian QFT, phase estimation, and discrete log

Quantum Fourier transform

Perhaps the most important unitary transformation in quantum computing is the *quantum Fourier transform* (QFT). Later, we will discuss the QFT over arbitrary finite groups, but for now we will focus on the case of an abelian group G. Here the transformation is

$$F_G := \frac{1}{\sqrt{|G|}} \sum_{x\in G} \sum_{y\in \hat{G}} \chi_y(x) |y\rangle\langle x| \tag{1}$$

where $\hat{G}$ is a complete set of characters of G, and $\chi_y(x)$ denotes the yth character of G evaluated at x. (You can verify that this is a unitary operator using the orthogonality of characters.) Since G and $\hat{G}$ are isomorphic, we can label the elements of $\hat{G}$ using elements of G, and it is often useful to do so.

The simplest QFT over a family of groups is the QFT over $G = \mathbb{Z}_2^n$. The characters of this group are $\chi_y(x) = (-1)^{x\cdot y}$, so the QFT is simply

$$F_{\mathbb{Z}_2^n} = \frac{1}{\sqrt{2^n}} \sum_{x,y\in\mathbb{Z}_2^n} (-1)^{x\cdot y} |y\rangle\langle x| = H^{\otimes n}. \tag{2}$$

You have presumably seen how this transformation is used in the solution of Simon's problem.

QFT over $\mathbb{Z}_{2^n}$

A more complex quantum Fourier transform is the QFT over $G = \mathbb{Z}_{2^n}$:

$$F_{\mathbb{Z}_{2^n}} = \frac{1}{\sqrt{2^n}} \sum_{x,y\in\mathbb{Z}_{2^n}} \omega_{2^n}^{xy} |y\rangle\langle x| \tag{3}$$

where $\omega_m := \exp(2\pi \mathrm{i}/m)$ is a primitive mth root of unity. To see how to realize this transformation by a quantum circuit, it is helpful to represent the input x as a string of bits, $x = x_{n-1}\dots x_1 x_0$, and to consider how an input basis vector is transformed:

$$|x\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{y\in\mathbb{Z}_{2^n}} \omega_{2^n}^{xy} |y\rangle \tag{4}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y\in\mathbb{Z}_{2^n}} \omega_{2^n}^{x(\sum_{k=0}^{n-1} y_k 2^k)} |y_{n-1}\dots y_1 y_0\rangle \tag{5}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y\in\mathbb{Z}_{2^n}} \prod_{k=0}^{n-1} \omega_{2^n}^{x y_k 2^k} |y_{n-1}\dots y_1 y_0\rangle \tag{6}$$

$$= \frac{1}{\sqrt{2^n}} \bigotimes_{k=0}^{n-1} \sum_{y_k\in\mathbb{Z}_2} \omega_{2^n}^{x y_k 2^k} |y_k\rangle \tag{7}$$

$$= \bigotimes_{k=0}^{n-1} |z_k\rangle \tag{8}$$

where

$$|z_k\rangle := \frac{1}{\sqrt{2}} \sum_{y_k \in \mathbb{Z}_2} \omega_{2^n}^{x y_k 2^k} |y_k\rangle \tag{9}$$

$$= \frac{1}{\sqrt{2}} (|0\rangle + \omega_{2^n}^{x 2^k} |1\rangle) \tag{10}$$

$$= \frac{1}{\sqrt{2}} (|0\rangle + \omega_{2^n}^{\sum_{j=0}^{n-1} x_j 2^{j+k}} |1\rangle) \tag{11}$$

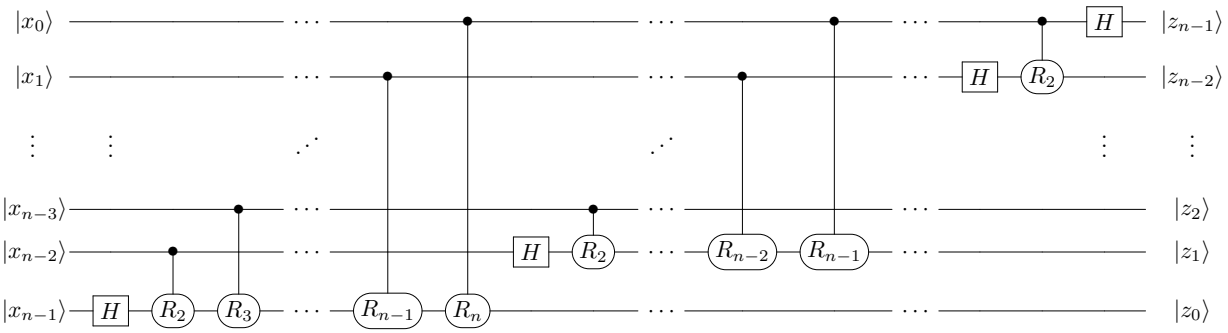
$$= \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi \mathrm{i}(x_0 2^{k-n} + x_1 2^{k+1-n} + \cdots + x_{n-1-k} 2^{-1})} |1\rangle). \tag{12}$$

(A more succinct way to write this is $|z_k\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \omega_{2^{n-k}}^{x} |1\rangle)$, but the above expression is more helpful for understanding the circuit.) In other words, $F|x\rangle$ is a tensor product of single-qubit states, where the kth qubit only depends on the k least significant bits of x.

This decomposition immediately gives a circuit for the QFT over $\mathbb{Z}_{2^n}$. Let R_k denote the single-qubit unitary operator

$$R_k := \begin{pmatrix} 1 & 0 \\ 0 & \omega_{2^k} \end{pmatrix}. \tag{13}$$

Then the circuit can be written as follows:



This circuit uses $O(n^2)$ gates. However, there are many rotations by small angles that do not affect the final result very much. If we simply omit the gates R_k with $k = \Omega(\log n)$, then we obtain a circuit with $O(n \log n)$ gates that implements the QFT with precision $1/\operatorname{poly}(n)$.

Phase estimation

Aside from being directly useful in quantum algorithms, such as Shor's algorithm, The QFT over $\mathbb{Z}_{2^n}$ provides a useful quantum computing primitive called *phase estimation*. In the phase estimation problem, we are given a unitary operator U (either as an explicit circuit, or as a black box that lets us apply a controlled-U^j operation for integer values of j). We are also given a state $|\phi\rangle$ that is promised to be an eigenvector of U, namely $U|\phi\rangle = e^{\mathrm{i}\phi}|\phi\rangle$ for some $\phi \in \mathbb{R}$. The goal is to output an estimate of ϕ to some desired precision.

The procedure for phase estimation is straightforward. To get an n-bit estimate of ϕ, prepare the quantum computer in the state

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_{2^n}} |x, \phi\rangle, \tag{14}$$

apply the operator

$$\sum_{x \in \mathbb{Z}_{2^n}} |x\rangle\langle x| \otimes U^x \tag{15}$$

to give the state

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_{2^n}} e^{\mathrm{i}\phi x} |x, \phi\rangle, \tag{16}$$

apply an inverse Fourier transform on the first register, and measure. If the binary expansion of $\phi/2\pi$ terminates after at most n bits (i.e., if $\phi = 2\pi y/2^n$ for some $y \in \mathbb{Z}_{2^n}$), then the state (16) is $F_{2^n}|y\rangle \otimes |\phi\rangle$, so the result is guaranteed to be the binary expansion of $\phi/2\pi$. In general, we obtain a good approximation with high probability. In particular, the probability of obtaining the result y (corresponding to the estimate $2\pi y/2^n$ for the phase) is

$$\Pr(y) = \frac{1}{2^{2n}} \cdot \frac{\sin^2(2^{n-1}\phi)}{\sin^2(\frac{\phi}{2} - \frac{\pi y}{2^n})}, \tag{17}$$

which is strongly peaked around the best n-bit approximation (in particular, it gives the best n-bit approximation with probability at least $4/\pi^2$). We will see the details of a similar calculation when we discuss period finding.

QFT over $\mathbb{Z}_N$ and over a general finite abelian group

One useful application of phase estimation is to implement the QFT over an arbitrary cyclic group $\mathbb{Z}_N$:

$$F_{\mathbb{Z}_N} = \frac{1}{\sqrt{N}} \sum_{x,y \in \mathbb{Z}_N} \omega_N^{xy} |y\rangle\langle x|. \tag{18}$$

The circuit we derived using the binary representation of the input and output only works when N is a power of two (or, with a slight generalization, some other small integer). But there is a simple way to realize $F_{\mathbb{Z}_N}$ (approximately) using phase estimation.

We would like to perform the transformation that maps $|x\rangle \mapsto |\tilde{x}\rangle$, where $|\tilde{x}\rangle := F_{\mathbb{Z}_N}|x\rangle$ denotes a Fourier basis state. (By linearity, if the transformation acts correctly on a basis, it acts correctly on all states.) It is straightforward to perform the transformation $|x, 0\rangle \mapsto |x, \tilde{x}\rangle$; then it remains to erase the register $|x\rangle$ from such a state.

Consider the unitary operator that adds 1 modulo N:

$$U := \sum_{x \in \mathbb{Z}_N} |x+1\rangle\langle x|. \tag{19}$$

The eigenstates of this operator are precisely the Fourier basis states $|\tilde{x}\rangle := F_{\mathbb{Z}_N}|x\rangle$, since (as a simple calculation shows)

$$F_{\mathbb{Z}_N}^\dagger U F_{\mathbb{Z}_N} = \sum_{x \in \mathbb{Z}_N} \omega_N^x |x\rangle\langle x|. \tag{20}$$

Thus, using phase estimation on U (with n bits of precision where $n = O(\log N)$), we can perform the transformation

$$|\tilde{x}, 0\rangle \mapsto |\tilde{x}, x\rangle \tag{21}$$

(actually, phase estimation only gives an approximation of x, so we implement this transformation only approximately). By running this operation in reverse, we can erase $|x\rangle$, and thereby produce the desired QFT.

Given the Fourier transform over $\mathbb{Z}_N$, it is straightforward to implement the QFT over an arbitrary finite abelian group: any finite abelian group can be written as a direct product of cyclic factors, and the QFT over a direct product of groups is simply the tensor product of QFTs over the individual groups.

Discrete log

One of the applications of the QFT over a cyclic group is to the solution of the discrete log problem. This problem is defined as follows. Let $G = \langle g \rangle$ be a cyclic group generated by g, written multiplicatively. Given an element $x \in G$, the *discrete logarithm of x in G with respect to g*, denoted $\log_g x$, is the smallest non-negative integer α such that $g^\alpha = x$. The *discrete logarithm problem* is the problem of calculating $\log_g x$.

Here are some simple examples of discrete logarithms:

- For any $G = \langle g \rangle$, $\log_g 1 = 0$
- For $G = \mathbb{Z}_7^\times$, $\log_3 2 = 2$
- For $G = \mathbb{Z}_{541}^\times$, $\log_{126} 282 = 101$

The discrete logarithm seems like a good candidate for a one-way function. We can efficiently compute g^α, even if α is exponentially large (in $\log |G|$), using repeated squaring. But given x, it is not immediately clear how to compute $\log_g x$, other than by checking exponentially many possibilities. (There are better algorithms than brute force search, but none is known that works in polynomial time.) This is the basis of the well-known Diffie-Hellman key exchange protocol.

Shor's algorithm

Now we will see how Shor's algorithm can be used to calculate discrete logarithms. This is a nice example because it's simpler than the factoring algorithm, but the problem it solves is actually at least as hard: factoring N can be reduced to calculating discrete log in $\mathbb{Z}_N^\times$. (Unfortunately, this does not by itself give a quantum algorithm for factoring, because Shor's algorithm for discrete log in G requires us to know the order of G—but computing $|\mathbb{Z}_N^\times| = \phi(N)$ is as hard as factoring N.)

Given some element x of a cyclic group $G = \langle g \rangle$, we would like to calculate $\log_g x$, the smallest integer α such that $g^\alpha = x$. For simplicity, let us assume that the order of G, $N := |G|$, is known. (For example, if $G = \mathbb{Z}_p^\times$ for prime p, then we know $N = p - 1$. In general, if we do not know N, we can learn it using Shor's period-finding algorithm, which we'll review later.) We can also assume that $x \neq g$ (i.e., $\log_g x \neq 1$), since it is easy to check whether this is the case.

To approach this problem, consider the function $f \colon \mathbb{Z}_N \times \mathbb{Z}_N \to G$ as follows:

$$f(\alpha, \beta) = x^\alpha g^\beta. \tag{22}$$

Since $f(\alpha, \beta) = g^{\alpha \log_g x + \beta}$, f is constant on the lines

$$L_\gamma := \{(\alpha, \beta) \in \mathbb{Z}_N^2 \colon \alpha \log_g x + \beta = \gamma\}. \tag{23}$$

Shor's algorithm for finding $\log_g x$ proceeds as follows. We start from the uniform superposition over $\mathbb{Z}_N \times \mathbb{Z}_N$ and compute the hiding function in another register:

$$|\mathbb{Z}_N \times \mathbb{Z}_N\rangle := \frac{1}{N} \sum_{\alpha,\beta \in \mathbb{Z}_N} |\alpha, \beta\rangle \mapsto \frac{1}{N} \sum_{\alpha,\beta \in \mathbb{Z}_N} |\alpha, \beta, f(\alpha, \beta)\rangle. \tag{24}$$

Then we discard the third register. To see what this does, it may be conceptually helpful to imagine that we actually measure the third register. Then the post-measurement state is a superposition over group elements consistent with the observed function value (say, g^δ), which is simply the set of points on some line L_δ. In other words, we get the state

$$|L_\delta\rangle = \frac{1}{\sqrt{N}} \sum_{\alpha \in \mathbb{Z}_N} |\alpha, \delta - \alpha \log_g x\rangle \tag{25}$$

However, note that the measurement outcome is unhelpful: each possible value occurs with equal probability, and we cannot obtain δ from g^δ unless we know how to take discrete logarithms. Thus we may as well simply discard the third register, leaving the system in the mixed state described by the ensemble of pure states (25) where δ is uniformly random and unknown.

Now we can exploit the symmetry of the quantum state by performing a QFT over $\mathbb{Z}_N \times \mathbb{Z}_N$; then the state becomes

$$\frac{1}{N^{3/2}} \sum_{\alpha,\mu,\nu \in \mathbb{Z}_N} \omega_N^{\mu\alpha + \nu(\delta - \alpha \log_g x)} |\mu, \nu\rangle = \frac{1}{N^{3/2}} \sum_{\mu,\nu \in \mathbb{Z}_N} \omega_N^{\nu\delta} \sum_{\alpha \in \mathbb{Z}_N} \omega_N^{\alpha(\mu - \nu \log_g x)} |\mu, \nu\rangle, \tag{26}$$

and using the identity $\sum_{\alpha \in \mathbb{Z}_N} \omega_N^{\alpha\beta} = N\delta_{\beta,0}$, we have

$$\frac{1}{\sqrt{N}} \sum_{\nu \in \mathbb{Z}_N} \omega_N^{\nu\delta} |\nu \log_g x, \nu\rangle. \tag{27}$$

Now suppose we measure this state in the computational basis. Then we obtain some pair $(\nu \log_g x, \nu)$ for uniformly random $\nu \in \mathbb{Z}_N$. If ν has a multiplicative inverse modulo N, we can divide the first register by ν to get the desired answer. If ν does not have a multiplicative inverse, we simply repeat the entire procedure again. The probability of success for each independent attempt is $\phi(N)/N = \Omega(1/\log\log N)$ (where $\phi(N)$ denotes the number of positive integers less than and relatively prime to n), so we don't have to repeat the procedure many times before we find an invertible ν.

This algorithm can be carried out for any cyclic group G so long as we have a unique representation of the group elements, and we are able to efficiently compute products in G. (We need to be able to compute high powers of a group element, but recall that this can be done quickly by repeated squaring.) In particular, it can also be used to solve the discrete log problem for elliptic curves, thus compromising most elliptic curve cryptography.