

Quantum Fourier Transform

This lecture will concentrate almost entirely upon a single unitary transformation: the *quantum Fourier transform*. This is a discrete Fourier transform, not upon the data stored in the system state, but upon the state itself.

Let's look at the definition to make this a bit clearer. The discrete Fourier transform (DFT) of a discrete function f_1, \dots, f_N is given by

$$\tilde{f}_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i j k / N} f_j.$$

The inverse transform is

$$f_j \equiv \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-2\pi i j k / N} \tilde{f}_k.$$

In the *quantum* Fourier transform, we do a DFT on the *amplitudes* of a quantum state:

$$\sum_j \alpha_j |j\rangle \rightarrow \sum_k \tilde{\alpha}_k |k\rangle,$$

where

$$\tilde{\alpha}_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i j k / N} \alpha_j.$$

The question is: can we actually carry out this transform physically? This would be possible if there were a unitary operator \hat{F} which transformed a state into its DFT:

$$|\tilde{\psi}\rangle = \hat{F}|\psi\rangle,$$

$$\hat{F}^\dagger \hat{F} = \hat{I}.$$

First, we observe that the amplitudes $\tilde{\alpha}_k$ are linear in the original α_j . So there is a linear operator \hat{F} which implements the transform. We can write it in outer product notation:

$$\hat{F} = \sum_{j,k=0}^{N-1} \frac{e^{2\pi ijk/N}}{\sqrt{N}} |k\rangle\langle j|.$$

It is easy to check that this does indeed produce the correct transformation

$$|\psi\rangle \rightarrow |\tilde{\psi}\rangle.$$

All that remains is to check that \hat{F} is unitary. Plugging in the above definition,

$$\hat{F}^\dagger = \left(\sum_{j,k} \frac{e^{-2\pi i j k / N}}{\sqrt{N}} |j\rangle \langle k| \right),$$

$$\hat{F} = \left(\sum_{j',k'} \frac{e^{2\pi i j' k' / N}}{\sqrt{N}} |k'\rangle \langle j'| \right),$$

$$\begin{aligned} \hat{F}^\dagger \hat{F} &= \frac{1}{N} \sum_{j,k,j',k'} e^{2\pi i (j' k' - j k) / N} |j\rangle \langle j'| \delta_{kk'} \\ &= \frac{1}{N} \sum_{j,k,j'} e^{2\pi i (j' - j) k / N} |j\rangle \langle j'| \\ &= \sum_{j,j'} |j\rangle \langle j'| \delta_{jj'} \\ &= \sum_j |j\rangle \langle j| = \hat{I}. \end{aligned}$$

So \hat{F} is unitary.

The Fourier transform lets us define a new basis: $|\tilde{x}\rangle = \hat{F}|x\rangle$, where $\{|x\rangle\}$ is the usual computational basis. This basis has a number of interesting properties.

Every vector $|\tilde{x}\rangle$ is an equally weighted superposition of all the computational basis states:

$$\begin{aligned} |\langle \tilde{x}|y\rangle|^2 &= \langle y|\tilde{x}\rangle \langle \tilde{x}|y\rangle \\ &= \langle y|\hat{F}|x\rangle \langle x|\hat{F}^\dagger|y\rangle \\ &= \frac{e^{2\pi ixy/N}}{\sqrt{N}} \frac{e^{-2\pi ixy/N}}{\sqrt{N}} = \frac{1}{N}. \end{aligned}$$

So if we think of the states $|x\rangle$ as being somehow the most “classical,” then the states $|\tilde{x}\rangle$ are somehow as “unclassical” as possible.

Recall that the Hadamard transform could also turn computational basis states into equally weighted superpositions of all states. But it left all amplitudes real, while the amplitudes of $|\tilde{x}\rangle$ are complex. And it was its own inverse, while $\hat{F} \neq \hat{F}^\dagger$.

From the point of view of physics, the relationship of this basis to the computational basis is analogous to that between the *momentum* and *position* bases of a particle. One way to see this is to note how the *generator of translations* looks in both bases.

By the generator of translations I mean the unitary operator \hat{S} which changes the value of x by 1:

$$\hat{S}|x\rangle = |(x + 1) \bmod N\rangle.$$

What effect does this have on the basis $\{|\tilde{x}\rangle\}$? It is easy to check that

$$\hat{S}|\tilde{x}\rangle = e^{-2\pi ix/N}|\tilde{x}\rangle.$$

A translation in x produces a phase in \tilde{x} . (The reverse is also true.) This is like the translation operators for continuous particles.

Circuits for the Fourier Transform

At this point we will specialize to the case of n q-bits, so the dimension is $N = 2^n$.

We have seen that the quantum Fourier transform is a unitary operator. Therefore, by our earlier results, there is a quantum circuit which implements it. However, there is no guarantee that this circuit will be efficient; a general unitary requires a circuit with a number of gates exponential in the number of bits.

Very fortunately, in this case an efficient circuit does exist. (Fortunate, because the Fourier transform is at the heart of the most impressive quantum algorithms!)

The key insight into designing a circuit for the Fourier transform is to notice that the states $|\tilde{j}\rangle$ can be written in a product form.

Let the binary expression for j be $j_1j_2 \dots j_n$, where $j = j_12^{n-1} + j_22^{n-2} + \dots + j_n$.

We also write binary fractions $0.j_1j_2 \dots j_n = j_1/2 + j_2/4 + \dots + j_n/2^n = j/2^n$.

Then

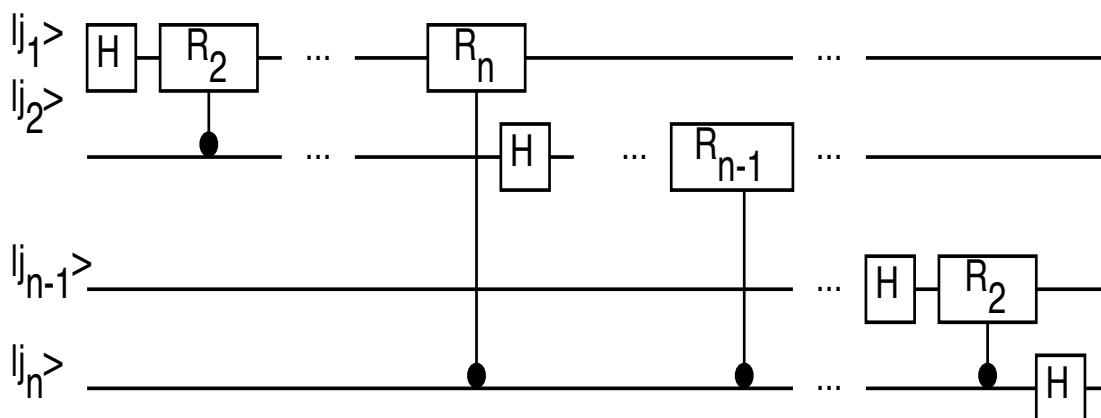
$$\begin{aligned} |\tilde{j}\rangle = & 2^{-n/2} (|0\rangle + e^{2\pi i 0.j_n} |1\rangle) \\ & \otimes (|0\rangle + e^{2\pi i 0.j_{n-1}j_n} |1\rangle) \\ & \otimes \dots \otimes (|0\rangle + e^{2\pi i 0.j_1j_2 \dots j_n} |1\rangle). \end{aligned}$$

The unitary $|0, 1\rangle \rightarrow (|0\rangle \pm \exp(i\theta)|1\rangle)/\sqrt{2}$ is a Hadamard followed by a rotation of $\theta/2$ around the Z axis. In the expression above, the rotation depends on the values of the other bits. So we should expect to be able to build the Fourier transform out of Hadamards and controlled phase rotation gates.

Define the rotation

$$\hat{R}_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}.$$

The controlled- \hat{R}_k gate performs this if and only if a control bit is $|1\rangle$ rather than $|0\rangle$. Putting these together with the Hadamards gives the following circuit:



This circuit performs the Fourier transform with the bits of the transformed state in *reverse* order. This circuit uses $n^2/2$ controlled-R gates, each of which can be produced with two CNOTs (as we saw last time). So the circuit as a whole uses n^2 CNOTS—definitely polynomial.

Periodic states

Suppose we are in N dimensions, and given a state of the form

$$|\phi\rangle = \sum_{n=0}^{N/r-1} c|\ell + nr\rangle,$$

where $|c| = \sqrt{r/N}$. We call this a *periodic* state with *period* r and *offset* ℓ .

What happens when we transform such a periodic state $|\phi\rangle \rightarrow |\tilde{\phi}\rangle$? The new state will have the form

$$|\tilde{\phi}\rangle = \sum_{m=0}^{r-1} \alpha_m |mN/r\rangle,$$

where $|\alpha_m| = \sqrt{1/r}$ for all m .

This state is also periodic. The α_m can have nontrivial phases, but they are all of equal weight; and the offset is *zero*.

Period finding

We can exploit this fact to produce a quantum algorithm for *period finding*. Suppose $f(x)$ is a function from n -bit numbers to m -bit numbers. We have two quantum registers, an n -bit input register and an m -bit output register, and we prepare them in the state

$$|\Psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{2^n-1} |x\rangle|0\rangle$$

using n Hadamard gates. (In general, there may be scratch bits as well, but we'll ignore that for now.)

We then apply a circuit that performs the unitary \hat{U}_f :

$$\hat{U}_f|\Psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle.$$

Suppose now that we measure the output register *only*, and get a particular value a . Then the input will be left in an evenly-weighted superposition of all x such that $f(x) = a$.

If $f(x)$ is a periodic function with period r , then this state will look like this:

$$\frac{1}{\sqrt{N/r}} \sum_{n=0}^{N/r-1} |x_0 + nr\rangle |a\rangle,$$

where x_0 is the smallest value for which $f(x_0) = a$. The input register is in a periodic state.

Let us Fourier transform the input register. Then we'll get a state of the form

$$\sum_{m=0}^{r-1} \alpha_m |mN/r\rangle |a\rangle,$$

where $|\alpha_m| = \sqrt{1/r}$ and the particular phases of the α_m will depend upon the measured value of a .

What would happen if we now measure the *input* register? We will get one value mN/r for some value of m between 0 and $r - 1$.

This by itself is not enough to tell us what N/r (and hence r) is. But let us run the algorithm d times. We will get a sequence of integers $m_1N/r, \dots, m_dN/r$ which are all multiples of N/r . For a number of runs d which grows only moderately in N , we can be confident that with high probability, N/r is the *only* common factor of all the numbers.

Please note that we have implicitly assumed that $f(x) = f(y)$ *only* if $x = y + nr$ —that is, except for the periodicity, this function has no repeated values. The reality can be more complicated, producing states which are superpositions of different periods with different weights. Fortunately, the functions we need for our algorithms have the simpler structure.

Greatest Common Divisor

Since the time of the ancient Greeks, an efficient algorithm has been known for finding the greatest common divisor (GCD) of two numbers: Euclid's algorithm.

Suppose a and b are both multiples of some common divisor n , with $a > b$. Then if I divide (say) a by b , the *remainder* $a \bmod b$ will also be a multiple of n , and smaller than either a or b .

We repeat this procedure, this time with b and $a \bmod b$, and so on, until we reach the point where one of our numbers divides the other exactly. This number is the GCD of a and b . Since the numbers get smaller each time, we obviously must eventually find the answer; and more detailed analysis shows that it is computationally efficient.

If we take our numbers $m_1N/r, \dots, m_dN/r$ and pairwise perform GCD on them, with high probability we will find the greatest common divisor of all of them to be N/r . This obviously gives us the value of r ; and we have found the period of $f(x)$ by an efficient quantum algorithm (assuming that \hat{U}_f can be done efficiently).

In fact, going back over the algorithm, we find that the measurement of the output register is not really necessary: because different values of $f(x)$ are orthogonal, the associated periodic states of the input register cannot interfere with each other. Surprisingly enough, once we have calculated $f(x)$ by applying \hat{U}_f , we have no further use for the output register; if we like, we can throw it away!

Finding the period of a function turns out to be the key to Shor's factoring algorithm, as well as a number of other efficient algorithms. But before we get to those, we will go back and study a much more sophisticated version of this technique.

Next time: the phase-estimation algorithm.