Quantum Fourier Transform

1 Fourier Transform

Definition 19.1 (FT_N): The Fourier transform mod N is the $N \times N$ matrix given by

$$FT_N = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)^2} \end{pmatrix} ,$$

where $\omega = e^{2\pi i/N}$ is a primitive Nth root of unity. That is, the i,j'th element of FT_N is $\frac{1}{\sqrt{N}}\omega^{ij}$, for $i,j = 0,\ldots,N-1$.

Equivalently, in ket notation, for $j \in \{0,1,\ldots,N-1\}$, $FT_N|j\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}\omega^{ij}|i\rangle$.

Lemma 19.1: *FT_N is unitary.*

Proof: We need to check the inner product between the ith and jth columns of FT_N, that $\langle i|FT_N^\dagger FT_N|j\rangle = \delta_{ij} \equiv \begin{cases} 1 \text{ if } i=j \\ 0 \text{ if } i \neq j \end{cases}$. Indeed, this inner product is

$$\frac{1}{N}\sum_{k=0}^{N-1}\overline{\omega^{ik}}\omega^{jk} = \frac{1}{N}\sum_{k=0}^{N-1}\omega^{k(j-i)} \quad .$$

This is a geometric series with ratio between terms ω^{j-i} and so can easily be evaluated. If $i = j \bmod N$, then each term is $\omega^0 = 1$, so the inner product is $N/N = 1$. If $i \neq j$, then the sum is

$$1 + \omega^{j-i} + \omega^{2(j-i)} + \cdots + \omega^{(N-2)(j-i)} + \omega^{(N-1)(j-i)} \quad .$$

Multiplying the sum by $\omega^{i-j} \neq 1$ gives

$$\omega^{j-i} + \omega^{2(j-i)} + \omega^{3(j-i)} + \cdots + \omega^{(N-1)(j-i)} + \omega^{N(j-i)} \quad .$$

But $\omega^{N(j-i)} = (\omega^N)^{j-i} = 1$, so we have just rearranged the terms of the summation; the sum itself doesn't change when multiplied by ω^{j-i}. Therefore the sum is zero, as claimed. □

The above calculation is important enough to be stated separately:

Lemma 19.2: *Let ω be a primitive Nth root of unity (i.e., $\omega^N = 1$ but $\omega^m \neq 1$ for $0 < m < N$). Then*

$$\sum_{k=0}^{N-1}\omega^{kj} = \begin{cases} N & \text{if } j = 0 \bmod N \\ 0 & \text{otherwise} \end{cases} \quad .$$

2 Fast Fourier Transform

Computing (classically) the discrete Fourier transform is essential for digital signal processing. In this case we have a list of N numbers which we wish to convert into the Fourier basis by multiplying by the FT_N

matrix. Naïve matrix multiplication however takes $\Theta(N^2)$ steps; we take the dot product of each of the N rows of FT_N with the input column vector, of length N.

In fact, the FT_N matrix has a nice structure which lets us apply the Fourier transform in only $O(N \log N)$ steps – an important (classical) algorithm known as the *Fast Fourier Transform*.

How does the Fast Fourier Transform work? We'll do an example with $N = 6$. The Fourier transform matrix is

$$FT_6 = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 \\ 1 & \omega^2 & \omega^4 & 1 & \omega^2 & \omega^4 \\ 1 & \omega^3 & 1 & \omega^3 & 1 & \omega^3 \\ 1 & \omega^4 & \omega^2 & 1 & \omega^4 & \omega^2 \\ 1 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix} .$$

(Here, we have used that $\omega^6 = 1$. So for example the bottom right entry is $\omega^{5 \cdot 5} = (\omega^6)^4 \omega = \omega$.) Notice that the blue-shaded entries, put together, give the matrix

$$\frac{1}{\sqrt{6}} \begin{pmatrix} 1 & 1 & 1 \\ 1 & \omega^2 & \omega^4 \\ 1 & \omega^4 & \omega^2 \end{pmatrix} .$$

Since $\omega_6^2 = e^{\frac{2\pi i}{6} 2} = e^{\frac{2\pi i}{3}} = \omega_3$ a primitive third root of unity, the above matrix is just $\sqrt{2}$ times FT_3. The red-shaded entries of FT_6 also give $\sqrt{2}$ times FT_3. The green-shaded entries give $\sqrt{2} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \omega & 0 \\ 0 & 0 & \omega^2 \end{pmatrix} FT_3$, and the unshaded entries give minus this (since $\omega_6^3 = e^{i\pi} = -1$).

Thus to apply FT_6 to some vector, we need only apply FT_3 to its even entries (indexed 0, 2 and 4), and FT_3 to its odd entries (indexed 1, 3 and 5), then add the results up in a certain manner. Since we use the first Fourier transform twice (blue and red entries), and also the second Fourier transform twice (green and white entries), this ends up saving us work.

The same procedure works in general, as long as N is even. If N is a power of two, $N = 2^n$, then the procedure can be applied recursively. Let $T(M)$ be the time it takes to apply FT_M. Then we find the recursion

$$T(N) = 2T(\tfrac{N}{2}) + O(N) \ ,$$

since we need to compute two Fourier transforms on half the dimension, then recombine the results. A solution to this recursion is $T(N) = O(N \log N)$. Indeed, substituting this in,

$$N \log_2 N \stackrel{?}{=} 2(\tfrac{N}{2} \log_2 \tfrac{N}{2}) + N$$

holds, since $\log_2 \frac{N}{2} = \log_2 N - 1$.

3 Classical vs. Quantum Fourier Transform

We just showed how to compute the Fourier transform of N numbers in $O(N \log N)$ time. Now consider these $N = 2^n$ numbers as determining a quantum state on $n = \log_2 N$ qubits. We will find a quantum circuit computing the Fourier transform:

$$QFT_N \left(\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \right) = \sum_{y \in \{0,1\}^n} \left(\tfrac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} \alpha_x \omega_N^{xy} \right) |y\rangle \ .$$

(Notice that we can consider x and y as either n-bit strings or as numbers between 0 and $N-1$ interchangeably; the bit string is the number in binary. So ω^{xy} makes sense.) Since we showed that FT_N is unitary, some

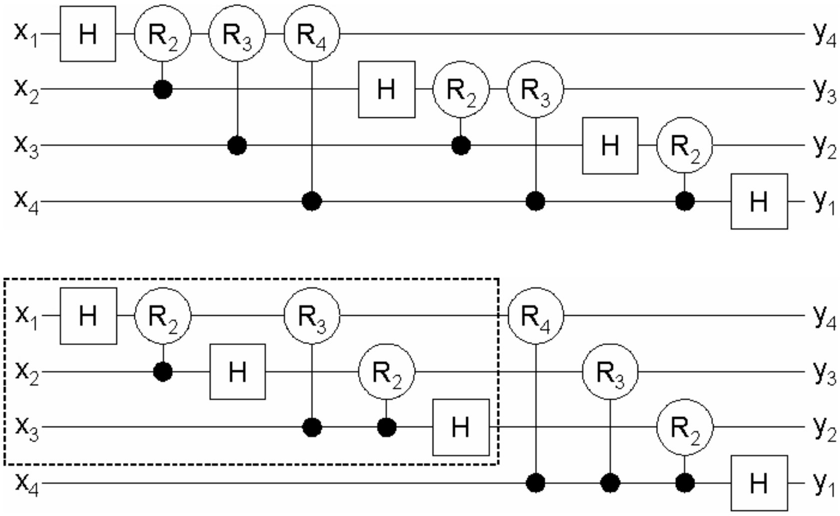


Figure 1: Two equivalent circuits each computing the Fourier transform mod 16. The boxed part of the second circuit computes FT_8.

quantum circuit computing it exists, but such a circuit could possibly be inefficient (take exponential time in n). In fact, our QFT circuit will take time $O(n^2) = O(\log^2 N)$. This is an exponential speedup over the classical Fast Fourier Transform!

But there's a catch. When we take the classical Fourier transform of $(\alpha_0, \ldots, \alpha_{N-1})$, we get the whole list $(\frac{1}{\sqrt{N}} \sum_x \alpha_x, \ldots, \frac{1}{\sqrt{N}} \sum_x \omega^{(N-1)x} \alpha_x)$, whereas quantumly the Fourier transform of $\sum_x \alpha_x |x\rangle$ gives $\frac{1}{\sqrt{N}} \sum_{x,y \in \{0,1\}^n} \omega_N^{xy} |y\rangle$. We cannot read off the list of the coefficients of a quantum state! Instead, we can (for example) measure the state, measuring y with probability

$$\frac{1}{N} |\sum_x \alpha_x \omega^{xy}|^2 \quad .$$

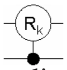
That is, we can *sample* from the Fourier-transformed coefficients. However, even with this limitation, the quantum Fourier transform is quite powerful.

4 Circuit for Quantum Fourier Transform

We'll show the circuit for the quantum Fourier transform mod $N = 2^n = 16$. It generalizes naturally. Let

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \omega_{2^k} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \omega_{2^n}^{2^{n-k}} \end{pmatrix} \quad ,$$

a phase shift. Two equivalent circuits computing the Fourier transform mod 16 are given in Figure 1. Here $x = x_1 x_2 x_3 x_4$ considered as a binary string, or $x = 2^3 x_1 + 2^2 x_2 + 2x_3 + x_4$ considered as a number in $\{0, 1, \ldots, 15\}$. Similarly for y.

The two-qubit gate  is a controlled-R_k gate; if the second (control) bit is 0, then nothing happens, and if it is 1, then R_k is applied to the first (target) bit. The matrix for this gate is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/2^k} \end{pmatrix} \quad .$$

That is, a phase of $e^{2\pi i/2^k}$ is added iff both bits are 1 (in this case, it actually doesn't matter which bit is considered the control, and which the target). We write

$$ {}^cR_k|a_0\rangle|a_1\rangle = e^{\frac{2\pi i}{2^k}a_0a_1} = \omega_N^{2^{n-k}a_0a_1} \quad . $$

Why do these circuits achieve the Fourier transform mod 16? Let U be the unitary transformation achieved by these circuits. We claim that for all $x, y \in \{0,1\}^n$, $\langle y|U|x\rangle = \langle y|FT_N|x\rangle$, so $U = FT_N$.

Proof: Indeed, because of the n Hadamard gates, U acting on $|x\rangle$ outputs an equal superposition over all $|y\rangle$

$$ U|x\rangle = \tfrac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{i\lambda_y}|y\rangle \quad , $$

except with some phase λ_y on $|y\rangle$. We need to show that

$$ \begin{aligned} \lambda_y &\overset{?}{=} 2\pi xy/N \\ &= \frac{2\pi}{2^n}\left(\sum_{h=0}^{n-1} 2^h x_{n-h}\right)\left(\sum_{i=0}^{n-1} 2^i y_{n-i}\right) \\ &\equiv \frac{2\pi}{2^n}\sum_{j=0}^{n-1} 2^j (x_{n-j}y_n + x_{n-j+1}y_{n-1} + \cdots + x_n y_{n-j}) \bmod 2\pi \\ &= \frac{2\pi}{2^n}\sum_{k=1}^{n} 2^{n-k}(x_k y_n + x_{k+1}y_{n-1} + \cdots + x_n y_k) \quad , \end{aligned} \tag{1} $$

where we have expanded out and multiplied the binary representations of x and y, dropping all terms with an integer multiple of 2π (this is why the sum over j only goes to $j = n-1$), and let $k = n - j$.

Indeed, consider the first circuit in Figure 1. A Hadamard gate on input wire l (output wire $n-l+1$) contributes a phase of -1 iff x_{n-l+1} and y_l are both 1 – that is, a phase of $\pi x_l y_{n-l+1} = 2\pi x_l y_{n-l+1}\frac{2^{n-1}}{2^n}$. Thus the total phase due to Hadamard gates is

$$ \tfrac{2\pi}{2^n} 2^{n-1} \sum_{l=1}^{n} x_l y_{n-l+1} \quad . $$

This is the $k = 1$ term of the sum in Eq. (1).

For $k > 1$, the ${}^cR_k$ gates act between the lth input wire and the $(n-l+k)$th input wire, $l = k, \ldots, n$, contributing a total phase

$$ \frac{2\pi}{2^n}\sum_{l=k}^{n} 2^{n-k} x_l y_{n-l+k} \quad , $$

the kth term of Eq. (1). $\square$

Another way of arguing that these circuits achieve FT_N is by induction on n, noticing that in the second of the circuits, the Fourier transform mod $\frac{N}{2}$ appears (boxed). This $FT_{N/2}$ arises from the same structure of the FT_N matrix which allowed the Fast Fourier Transform to work by two multiplications by $FT_{N/2}$ – quantumly one might say that both Fourier transforms $FT_{N/2}$ are computed in parallel, with additional phases added depending on whether the row (x) was even or odd. The last bit x_n determines whether x is even or odd.

5 Properties of the Fourier Transform

- Since $FT_N = \frac{1}{\sqrt{N}} \sum_{x,y=0}^{N-1} \omega_N^{xy} |x\rangle\langle y|$ is unitary, its inverse is $FT_N^\dagger = \frac{1}{\sqrt{N}} \sum_{x,y=0}^{N-1} \omega_N^{-xy} |y\rangle\langle x|$. This is just another Fourier transform, but using $e^{-2\pi i/N}$ instead of $e^{2\pi i/N}$ as the primitive Nth root of unity.

- If T_l is translation by l, $T_l|x\rangle = |x + l \bmod N\rangle$, and P_k is a change of phase, $P_k|x\rangle = \omega^{xk}|x\rangle$, then $(FT_N)T_lP_k = P_lT_{-k}(FT_N)$. Indeed,

$$\begin{aligned} FT_N\left(\sum_x \alpha_x \omega^{xk}|x+l\rangle\right) &= \frac{1}{\sqrt{N}} \sum_{x,y} \alpha_x \omega^{xk} \omega^{(x+l)y} |y\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{x,y} \omega^{yl} \alpha_x \omega^{x(y+k)} |y\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{x,y} \omega^{(y-k)l} \alpha_x \omega^{xy} |y-k\rangle \ . \end{aligned}$$

  We say that the Fourier transform maps translations to phase shifts, and vice versa. A phase shift doesn't change the probability of measuring a state in the standard basis – $|\langle y|P_k|\psi\rangle|^2 = |\omega^{-ky}\langle y|\Psi\rangle|^2 = |\langle y|\Psi\rangle|^2$. Thus for Fourier sampling, we find the interesting consequence that translating the input (cyclically) doesn't affect the output distribution.

- Let r divide N. Then

$$\begin{aligned} FT_N \frac{1}{\sqrt{N/r}} \sum_{j=0}^{N/r-1} |rj\rangle &= \frac{\sqrt{r}}{N} \sum_{y=0}^{N-1} \left(\sum_{j=0}^{N/r-1} \omega_N^{rjy} \right) |y\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{i=0}^{r-1} |\tfrac{N}{r} i\rangle \ , \end{aligned}$$

  where in the last step we used that $\omega_N^r = \omega_{N/r}$, so $\sum_{j=0}^{N/r-1} \omega_N^{rjy}$ is N/r if $y = 0 \bmod N/r$ and 0 otherwise. By the property $(FT_N)T_lP_k = P_lT_{-k}(FT_N)$, we see that, when $r|N$,

$$FT_N \frac{1}{\sqrt{N/r}} \sum_{j=0}^{N/r-1} \omega^{krj} |rj+l\rangle = \frac{1}{\sqrt{r}} \sum_{i=0}^{r-1} \omega^{l(\frac{N}{r}i-k)} |\tfrac{N}{r}i - k\rangle \ .$$

- In particular ($r = 1$), $FT_N|0\rangle = \frac{1}{\sqrt{N}} \sum_y |y\rangle$ the uniform superposition, and $FT_N \frac{1}{\sqrt{N}} \sum_y |y\rangle = |0\rangle$. This is just as for the n-fold Hadamard transform $H^{\otimes n}$. However, while the Fourier transform $\bmod 2$ is $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = H$, $FT_{2^n} \neq H^{\otimes n}$ for $n > 1$.

- We have shown how to compute the quantum Fourier transform $\bmod N$, where N is a power of two. Similar constructions work for computing the quantum Fourier transform $\bmod N$, where N is a power of any prime. For more general N: if r and s are relatively prime, we can combine the Fourier transforms $\bmod r$ and $\bmod s$ to yield a Fourier transform $\bmod rs$. Indeed, the Chinese Remainder Theorem gives a ring isomorphism between $\mathbf{Z}_{rs}$ and $\mathbf{Z}_r \times \mathbf{Z}_s$, given by

$$m \bmod (rs) = m_r s(s^{-1} \bmod r) + m_s r(r^{-1} \bmod s) \longleftrightarrow (m_r, m_s) \qquad \text{(where } m_p = m \bmod p\text{)} \ .$$

  Both directions of the isomorphism can be efficiently computed classically. By homework 7, there is a n efficient quantum circuit taking $|m\rangle$ to $|m \bmod r\rangle \otimes |m \bmod s\rangle$. Apply this circuit, then compute the quantum Fourier transform $\bmod r$ on the first register, and FT_s on the second register, and finally invert the isomorphism. Altogether, we get

$$|m\rangle \mapsto \frac{1}{\sqrt{rs}} \sum_{x=0}^{r-1} \sum_{y=0}^{s-1} \omega_r^{m_r x} \omega_s^{m_s y} |xs(s^{-1})_r + yr(r^{-1})_s\rangle \ .$$

This is not the Fourier transform $\bmod (rs)$, but it is close. With a little thought, we see that we should have applied the Fourier transform $\bmod r$ using $\omega_r^{s^{-1} \bmod r}$ as the primitive rth root of unity instead of just ω_r, and similarly we should have used $\omega_s^{r^{-1} \bmod s}$ as the primitive sth root of unity. Then

$$\begin{aligned}
|m\rangle &\mapsto \frac{1}{\sqrt{rs}} \sum_{xy} (\omega_r^{(s^{-1})_r})^{m_r x} (\omega_s^{(r^{-1})_s})^{m_s y} |xs(s^{-1})_r + yr(r^{-1})_s\rangle \\
&= \frac{1}{\sqrt{rs}} \sum_{xy} \omega_{rs}^{m_r xs(s^{-1})_r + m_s yr(r^{-1})_s} |xs(s^{-1})_r + yr(r^{-1})_s\rangle \\
&= \frac{1}{\sqrt{rs}} \sum_{z=0}^{rs-1} \omega_{rs}^{mz} |z\rangle \ .
\end{aligned}$$