The Fast Fourier Transform

Suppose that $x(t)$ is a function that is periodic of period 2ℓ and that we only know its values at N equally spaced points (in each period of course). Precisely, let

$$x[n] = x\big(n\tfrac{2\ell}{N}\big) \qquad\qquad n = 0, 1, 2, \cdots, N-1$$

Then it is natural to approximate the k^{th} Fourier coefficient $\frac{1}{2\ell}\int_0^{2\ell} e^{-ik\pi t/\ell} x(t)\,dt$ by a Riemann sum in which $e^{-ik\pi t/\ell}x(t)$ is replaced by a function which takes the constant value $e^{-2\pi ikn/N}x[n]$ for all t obeying $|t - n\frac{2\ell}{N}| < \frac{\ell}{N}$. The sum is

$$\hat{x}[k] = \tfrac{1}{N}\sum_{n=0}^{N-1} e^{-2\pi ikn/N} x[n] \tag{D1}$$

Note that $\hat{x}[k+N] = \hat{x}[k]$. Hence $\hat{x}[k]$ has period N, just like $x[n]$. The vector $\big(\hat{x}[k]\big)_{k=0,1,2,\cdots,N-1}$ is called the discrete Fourier transform of the vector $\big(x[n]\big)_{n=0,1,2,\cdots,N-1}$. As we saw in class, we can recover the $x[n]$'s from the $\hat{x}[k]$'s by the inverse formula

$$x[n] = \sum_{k=0}^{N-1} e^{2\pi ikn/N}\hat{x}[k] \tag{D2}$$

We now turn to the problem of evaluating, by computer, the right hand sides of (D1,2). The two are very similar, so let's concentrate on (D2). Define the complex number

$$\omega_N = e^{2\pi i/N}$$

We may rewrite (D2) as

$$x[n] = \sum_{k=0}^{N-1} \omega_N^{nk}\hat{x}[k]$$

Note that $\omega_N^N = e^{2\pi i} = 1$, so that there are only N different integer powers of ω_N, namely ω_N^0, ω_N^1, $\cdots$, ω_N^{N-1} , and they can all be computed with a total of $N-2$ complex multiplications. For each n, evaluating the sum $\sum_{k=0}^{N-1}\omega_N^{nk}\hat{x}[k]$ takes N complex multiplications, so it looks like computing $x[n]$ for all N values of n takes on the order of N^2 multiplications. The fast fourier transform refers to a family of algorithms that generates all the $x[n]$'s using only $O(N\log_2 N)$ operations. They became widely known only following a paper of J. W. Cooley and J. W. Tukey in 1965, though Gauss had known about them as early as about 1805. They are not a required part of the course.

One such algorithm, due to Danielson and Lanczos in 1942, is based on the following observation. Suppose that N is even. Then

$$\begin{aligned}
x[n] &= \sum_{k=0}^{N-1} \omega_N^{nk}\hat{x}[k] \\
&= \sum_{\substack{k=0\\ k\ \text{even}}}^{N-2} \omega_N^{nk}\hat{x}[k] + \sum_{\substack{k=1\\ k\ \text{odd}}}^{N-1} \omega_N^{nk}\hat{x}[k] \\
&= \sum_{p=0}^{N/2-1} \omega_N^{n2p}\hat{x}[2p] + \sum_{q=0}^{N/2-1} \omega_N^{n(2q+1)}\hat{x}[2q+1] \text{ where } k = 2p \text{ in the first sum and } k = 2q+1 \text{ in the second} \\
&= \sum_{p=0}^{N/2-1} \omega_{N/2}^{np}\hat{x}[2p] + \omega_N^n \sum_{q=0}^{N/2-1} \omega_{N/2}^{nq}\hat{x}[2q+1] \text{ since } \omega_N^2 = \omega_{N/2} \\
&= x[n]^{(0)} + \omega_N^n x[n]^{(1)}
\end{aligned}$$

where

$$x[n]^{(0)} = \sum_{k=0}^{N/2-1} \omega_{N/2}^{nk} \hat{x}[2k]$$
$$x[n]^{(1)} = \sum_{k=0}^{N/2-1} \omega_{N/2}^{nk} \hat{x}[2k+1]$$

are discrete Fourier series with $N/2$ rather than N points.

If one iterates this $\log_2 N$ times, one gets to the trivial problem of finding the discrete Fourier transform with N replaced by one. To see the iteration in action, lets do it explicitly for $N = 8$. Then the above computation becomes

$$\begin{aligned} x[n] &= \hat{x}[0] + \omega_8^n \hat{x}[1] + \omega_8^{2n} \hat{x}[2] + \omega_8^{3n} \hat{x}[3] + \omega_8^{4n} \hat{x}[4] + \omega_8^{5n} \hat{x}[5] + \omega_8^{6n} \hat{x}[6] + \omega_8^{7n} \hat{x}[7] \\ &= \left(\hat{x}[0] + \omega_4^n \hat{x}[2] + \omega_4^{2n} \hat{x}[4] + \omega_4^{3n} \hat{x}[6]\right) + \omega_8^n \left(\hat{x}[1] + \omega_4^n \hat{x}[3] + \omega_4^{2n} \hat{x}[5] + \omega_4^{3n} \hat{x}[7]\right) \\ &= x[n]^{(0)} + \omega_8^n x[n]^{(1)} \end{aligned}$$

In the next iteration

$$\begin{aligned} x[n]^{(0)} &= \hat{x}[0] + \omega_4^n \hat{x}[2] + \omega_4^{2n} \hat{x}[4] + \omega_4^{3n} \hat{x}[6] \\ &= (\hat{x}[0] + \omega_2^n \hat{x}[4]) + \omega_4^n (\hat{x}[2] + \omega_2^n \hat{x}[6]) \\ &= x[n]^{(00)} + \omega_4^n x[n]^{(01)} \\ x[n]^{(1)} &= \hat{x}[1] + \omega_4^n \hat{x}[3] + \omega_4^{2n} \hat{x}[5] + \omega_4^{3n} \hat{x}[7] \\ &= (\hat{x}[1] + \omega_2^n \hat{x}[5]) + \omega_4^n (\hat{x}[3] + \omega_2^n \hat{x}[7]) \\ &= x[n]^{(10)} + \omega_4^n x[n]^{(11)} \end{aligned}$$

and in the final iteration

$$\begin{aligned} x[n]^{(00)} &= \hat{x}[0] + \omega_2^n \hat{x}[4] = x[n]^{(000)} + \omega_2^n x[n]^{(001)} \\ x[n]^{(10)} &= \hat{x}[1] + \omega_2^n \hat{x}[5] = x[n]^{(100)} + \omega_2^n x[n]^{(101)} \\ x[n]^{(01)} &= \hat{x}[2] + \omega_2^n \hat{x}[6] = x[n]^{(010)} + \omega_2^n x[n]^{(011)} \\ x[n]^{(11)} &= \hat{x}[3] + \omega_2^n \hat{x}[7] = x[n]^{(110)} + \omega_2^n x[n]^{(111)} \end{aligned}$$

Hence, for $N = 8$, the FFT computation would be the following. For each n, first set

$$\begin{aligned} x[n]^{(000)} &= \hat{x}[0] & x[n]^{(100)} &= \hat{x}[1] & x[n]^{(010)} &= \hat{x}[2] & x[n]^{(110)} &= \hat{x}[3] \\ x[n]^{(001)} &= \hat{x}[4] & x[n]^{(101)} &= \hat{x}[5] & x[n]^{(011)} &= \hat{x}[6] & x[n]^{(111)} &= \hat{x}[7] \end{aligned}$$

Note that the argument k of each $\hat{x}[k]$ when expressed in binary is exactly the mirror image of the argument n of the corresponding $x[n]$. For example $6 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 110$. When the order of these binary digits are reversed we get 011, the supersrcript in $x[n]^{(011)} = \hat{x}[6]$. Next compute

$$\begin{aligned} x[n]^{(00)} &= x[n]^{(000)} + \omega_2^n x[n]^{(001)} & x[n]^{(10)} &= x[n]^{(100)} + \omega_2^n x[n]^{(101)} \\ x[n]^{(01)} &= x[n]^{(010)} + \omega_2^n x[n]^{(011)} & x[n]^{(11)} &= x[n]^{(110)} + \omega_2^n x[n]^{(111)} \end{aligned}$$

and then compute

$$\begin{aligned} x[n]^{(0)} &= x[n]^{(00)} + \omega_4^n x[n]^{(01)} \\ x[n]^{(1)} &= x[n]^{(10)} + \omega_4^n x[n]^{(11)} \end{aligned}$$

and finally

$$x[n] = x[n]^{(0)} + \omega_8^n x[n]^{(1)}$$