MA 511, Session 21

The Fast Fourier Transform (FFT)

We have seen that the first step to compute the truncated Fourier series from a sampled digital signal

digital sampling of signal $\longrightarrow$ truncated Fourier series (in complex form)

$$y = \begin{pmatrix} y_0 \\ \cdot \\ \cdot \\ \cdot \\ y_{n-1} \end{pmatrix} \longrightarrow c = \begin{pmatrix} c_0 \\ \cdot \\ \cdot \\ \cdot \\ c_{n-1} \end{pmatrix},$$

as well as the recovery of a signal from its Fourier coefficients are very fast:

$$c = F^{-1}y, \qquad y = Fc,$$

since F and F^{-1} can be determined by just finding the powers of a primitive n-th root of unity.

However, in order for this method to be so good, the multiplications $F^{-1}y$ and Fc must be very fast. There is indeed a super fast algorithm when n is a power of 2.

Assume $n = 2^l$ for some positive integer l, and let $m = \frac{n}{2} = 2^{l-1}$. We now split c into its even and odd components, respectively indicated by superscripts prime and double prime:

$$c' = \begin{pmatrix} c_0 \\ c_2 \\ \cdot \\ \cdot \\ \cdot \\ c_{n-2} \end{pmatrix}, \qquad c'' = \begin{pmatrix} c_1 \\ c_3 \\ \cdot \\ \cdot \\ \cdot \\ c_{n-1} \end{pmatrix}.$$

The Fast Fourier Transform is an algorithm based on using 2 systems of half the size,

$$y' = F_m c', \quad y'' = F_m c'',$$

whose components can be combined to recover the components of the original system.

Let

$$\omega = e^{2\pi i/n}$$

be a primitive n-th root of unity. It follows that $\omega^2 = e^{4\pi i/n} = e^{2\pi i/m}$ is a primitive m-th root of unity, and also

$$y = \begin{pmatrix} y_0 \\ \cdot \\ \cdot \\ \cdot \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} y'_0 + y''_0 \\ y'_1 + \omega y''_1 \\ \cdot \\ \cdot \\ \cdot \\ y'_{m-1} + \omega^{m-1} y''_{m-1} \\ y'_0 - y''_0 \\ y'_1 - \omega y''_1 \\ \cdot \\ \cdot \\ \cdot \\ y'_{m-1} - \omega^{m-1} y''_{m-1} \end{pmatrix}.$$

Example: For $n = 2$ we have $\omega = e^{\pi i} = -1$ and

$$y = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} c_0 + c_1 \\ c_0 - c_1 \end{pmatrix}.$$

Example: For $n = 4$ we have $\omega = e^{\pi i/2} = i$ and

$$y' = \begin{pmatrix} y'_0 \\ y'_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} c'_0 \\ c'_1 \end{pmatrix} = \begin{pmatrix} c_0 + c_2 \\ c_0 - c_2 \end{pmatrix},$$

$$y'' = \begin{pmatrix} y''_0 \\ y''_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} c''_0 \\ c''_1 \end{pmatrix} = \begin{pmatrix} c_1 + c_3 \\ c_1 - c_3 \end{pmatrix}.$$

It follows that

$$y = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} c_0 + c_1 + c_2 + c_3 \\ c_0 + ic_1 - c_2 - ic_3 \\ c_0 - c_1 + c_2 - c_3 \\ c_0 - ic_1 - c_2 + ic_3 \end{pmatrix}$$

$$= \begin{pmatrix} y'_0 + y''_0 \\ y'_1 + \omega y''_1 \\ y'_0 - y''_0 \\ y'_1 - \omega y''_1 \end{pmatrix} = \begin{pmatrix} (c_0 + c_2) + (c_1 + c_3) \\ (c_0 - c_2) + i(c_1 - c_3) \\ (c_0 + c_2) - (c_1 + c_3) \\ (c_0 - c_2) - i(c_1 - c_3) \end{pmatrix}.$$

Example: Compute $y = F_4 c$ by FFT for

$$c = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}.$$

Solution: We let

$$c' = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad c'' = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad F_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

We quickly obtain

$$y' = F_2 \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad y'' = F_2 \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}.$$

We finally use $\omega = i$ to obtain

$$y = \begin{pmatrix} 0+2 \\ 0+i0 \\ 0-2 \\ 0-i0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ -2 \\ 0 \end{pmatrix}.$$