3.3). THE FAST FOURIER TRANSFORM

The ubiquity of DFT's in practice is because there is a fast algorithm for computing them.

Applying the DFT means multiplying by F_n^{-1}, which has no non-zero elements so seems to require $O(n^2)$ floating point operations. ← n multiplications for each entry

In fact it is possible to do the transformation in a clever order to reduce the operation count to $O(n \log n)$ — called the Fast Fourier Transform (FFT).

- was known to Gauss, but rediscovered by Cooley & Tukey (1965).
- led to revolution in electronic analysis, control systems and compression.
- in particular, can analyse stream data (on same timescale as data acquired).

The key idea is the following factorisation:

$$F_n = \begin{bmatrix} I_{n/2} & D_{n/2} \\ I_{n/2} & -D_{n/2} \end{bmatrix} \begin{bmatrix} F_{n/2} & 0 \\ 0 & F_{n/2} \end{bmatrix} P_n$$

← important feature: lots of zeros!

where

$$I_{n/2} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 \end{bmatrix}, \qquad D_{n/2} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \omega & & \vdots \\ \vdots & & \omega^2 & \\ & & & \ddots \\ 0 & \cdots & & \omega^{n/2-1} \end{bmatrix}$$

i.e. a single 1 in each row & column

and $F_{n/2}$ is the Fourier matrix of size $n/2 \times n/2$. The third matrix P_n is a permutation matrix that separates the incoming vector $\vec{c}$ into odd and even parts:

$$P_n \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} c_0 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_1 \\ c_3 \\ \vdots \\ c_{n-1} \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{even entries} \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{odd entries} \end{matrix}$$

Example: $n = 4 \Rightarrow \omega = e^{i\pi/2} = i$.

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}, \quad I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D_2 = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$\omega = e^{i\pi} = -1$

The permutation matrix is

$$P_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad \longrightarrow \quad P_4 \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_2 \\ c_1 \\ c_3 \end{bmatrix}$$

Verify:

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & i \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -i \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & i \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -i \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

$$= F_4.$$

Proof (in general) :- (assumes n is even).

Let $\omega_n = e^{2\pi i/n}$ (to keep track of which n we are talking about.

The j^{th} entry of $\underline{f} = F_n \underline{c}$ is

$$f_j = \sum_{k=0}^{n-1} \omega_n^{jk} c_k = \underbrace{\sum_{k=0}^{n/2-1} \omega_n^{j2k} c_{2k}}_{\text{even}} + \underbrace{\sum_{k=0}^{n/2-1} \omega^{j(2k+1)} c_{2k+1}}_{\text{odd}}$$

$$= \sum_{k=0}^{n/2-1} (\omega_n^2)^{jk} c_{2k} + (\omega_n)^j \sum_{k=0}^{n/2-1} (\omega_n^2)^{jk} c_{2k+1}$$

$$\omega_n^2 = \left(e^{2\pi i/n}\right)^2 = e^{2\pi i/(n/2)} = \omega_{n/2} \quad (*)$$

— Key fact that makes it work.

$$= \underbrace{\sum_{k=0}^{n/2-1} \omega_{n/2}^{jk} c_{2k}}_{\text{apply } F_{n/2} \text{ to even } c_k} + (\omega_n)^j \underbrace{\sum_{k=0}^{n/2-1} (\omega_{n/2})^{jk} c_{2k+1}}_{\text{apply } F_{n/2} \text{ to odd } c_k} .$$

known as a "twiddle factor" (!)

Noting that $-\omega_n^{j-n/2} = -\left(e^{\frac{2\pi i}{n}}\right)^{-n/2} \omega_n^j = -e^{-\pi i} \omega_n^j = \omega_n^j$, we see that this gives the matrix factorisation. ▨

The reduction from F_n to two $F_{n/2}$'s cuts the work (almost) in half. But to reduce it much further, we can apply the factorisation recursively, ie. replace $F_{n/2}$ by two $F_{n/4}$'s, etc:

$$F_n = \begin{bmatrix} I_{n/2} & D_{n/2} \\ I_{n/2} & -D_{n/2} \end{bmatrix} \begin{bmatrix} F_{n/2} & 0 \\ 0 & F_{n/2} \end{bmatrix} \begin{bmatrix} c_0, c_2, c_4, \dots \\ c_1, c_3, c_5, \dots \end{bmatrix}$$

$$\begin{bmatrix} F_{n/2} & 0 \\ 0 & F_{n/2} \end{bmatrix} = \begin{bmatrix} \begin{matrix} I_{n/4} & D_{n/4} \\ I_{n/4} & -D_{n/4} \end{matrix} & 0 \\ 0 & \begin{matrix} I_{n/4} & D_{n/4} \\ I_{n/4} & -D_{n/4} \end{matrix} \end{bmatrix} \begin{bmatrix} F_{n/4} & & & 0 \\ & F_{n/4} & & \\ & & F_{n/4} & \\ 0 & & & F_{n/4} \end{bmatrix} \begin{bmatrix} c_0, c_4, c_8, \dots \\ c_2, c_6, c_{10}, \dots \\ c_1, c_5, c_9, \dots \\ c_3, c_7, c_{11}, \dots \end{bmatrix} .$$

How many multiplications are used? Suppose $n = 2^m$. Without the FFT, we needed $n^2 = 4^m$ operations. Now we have m stages, from 2^m down to 2^0. Each stage has $n/2$ multiplications to assemble the outputs from diagonal D's. (at 1, no multiplications are needed since $F_1 = 1$).

Hence the operation count is

$$\frac{n}{2} m = \frac{n}{2} \log_2(n).$$

e.g. if $n = 2^{10} = 1024$, then $n^2 = 2^{20} \simeq 1M$,
while $\frac{n}{2} m = 5(1024)$. $\longrightarrow$ major saving for larger matrices!

- The same idea works for the inverse (see Problem Sheet 3).
- A quick way to find the order of the $\underline{c}$'s (after a full recursion) is to write the numbers $0, \dots, n-1$ in base 2 and reverse the order of their bits: (bit-reversal)

e.g. $n = 4$

c_0	00	$\xrightarrow{\text{reverse}}$	00	c_0
c_1	01		10	c_2
c_2	10		01	c_1
c_3	11		11	c_3 .