Numerical Analysis – Lecture 18[1]

Method 4.16 *(The Hockney method)* Solving $\nabla^2 u = f$ in an $m \times m$ grid with the 5-point formula, we have written the equations in the form $\mathcal{A}\boldsymbol{u} = \boldsymbol{b}$, where $\mathcal{A}$ is block-TST (with A along the main diagonal and I in the sub-and-superdiagonal). The spectrum of A is known, $A = QDQ$, where Q is orthogonal and involutory ($Q^2 = I$) and D is diagonal, $D = \operatorname{diag} \boldsymbol{d}$. The vectors $\boldsymbol{u}$ and $\boldsymbol{b}$ are partitioned accordingly. Set $\boldsymbol{v}_k = Q\boldsymbol{u}_k$, $\boldsymbol{c}_k = Q\boldsymbol{b}_k$, $k = 1, \ldots, m$, therefore

$$\begin{bmatrix} D & I & & 0 \\ I & D & I & \\ & \ddots & \ddots & \ddots \\ 0 & & I & D \end{bmatrix} \boldsymbol{v} = \boldsymbol{c}.$$

Let us by this stage reorder the grid *by rows, instead of by columns.*. In other words, we permute $\boldsymbol{v} \mapsto \tilde{\boldsymbol{v}}$, $\boldsymbol{c} \mapsto \tilde{\boldsymbol{c}}$, s.t. $\tilde{\boldsymbol{c}}_1$, say, is made out of the first components of $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m$, $\tilde{\boldsymbol{c}}_2$ out of the second components and so on. This results in

$$\begin{bmatrix} \Gamma_1 & O & & \\ O & \Gamma_2 & O & \\ & \ddots & \ddots & \ddots \\ & & O & \Gamma_m \end{bmatrix} \tilde{\boldsymbol{v}} = \tilde{\boldsymbol{c}}, \text{ where } \Gamma_l = \begin{bmatrix} d_l & 1 & & 0 \\ 1 & d_l & 1 & \\ & \ddots & \ddots & \ddots \\ 0 & & 1 & d_l \end{bmatrix}_{m \times m}, \quad l = 1, \ldots, m.$$

These are m *uncoupled* systems, $\Gamma_l \tilde{\boldsymbol{v}}_l = \tilde{\boldsymbol{c}}_l$, $l = 1, \ldots, m$. Being *tridiagonal*, each such system can be solved fast. Thus, the steps of the algorithm and their computational cost are

1. Form the products $\boldsymbol{c}_k = Q\boldsymbol{b}_k$, $k = 1, \ldots, m$ ..... $\mathcal{O}(m^3)$
2. Solve $m \times m$ tridiagonal systems $\Gamma_l \tilde{\boldsymbol{v}}_l = \tilde{\boldsymbol{c}}_l$, $l = 1, \ldots, m$ ..... $\mathcal{O}(m^2)$
3. Form the products $\boldsymbol{u}_k = Q\boldsymbol{v}_k$, $k = 1, \ldots, m$ ..... $\mathcal{O}(m^3)$

(permutations $\boldsymbol{c} \mapsto \tilde{\boldsymbol{c}}$ and $\tilde{\boldsymbol{v}} \mapsto \boldsymbol{v}$ are free: in practice, we store $\boldsymbol{c}$ etc. as a 2D matrix, rather than in a long vector).

Method 4.17 *Improved Hockney algorithm* We observe that *the computational bottleneck is to be found in the $2m$ matrix-vector products by the matrix Q.* Recall further that $Q_{k,l} = \sqrt{\frac{2}{m+1}} \sin \frac{\pi kl}{m+1}$, $k, l = 1, \ldots, m$. This special form lends itself to a considerable speedup in matrix multiplication. Before making the problem simpler, however, let us make it more complicated! We write a typical product in the form

$$\sum_{l=1}^{m} x_l \sin \frac{\pi kl}{m+1} = \operatorname{Im} \sum_{l=0}^{m} x_l \exp \frac{\pi \mathrm{i} kl}{m+1} = \operatorname{Im} \sum_{l=0}^{2m+1} x_l \exp \frac{2\pi \mathrm{i} kl}{2m+2}, \quad k = 1, \ldots, m, \tag{4.8}$$

where $x_{m+1} = \cdots = x_{2m+1} = 0$.

Problem 4.18 *(The discrete Fourier transform)* Let Π_n be the space of all *bi-infinite complex n-periodic sequences: $\boldsymbol{x} = \{x_k\}_{k \in \mathbb{Z}} \in \Pi_n \Leftrightarrow x_{k+n} = x_k$, $k \in \mathbb{Z}$.* Set $\omega_n = \exp \frac{2\pi \mathrm{i}}{n}$, the primitive root of unity of degree n. The *discrete Fourier transform (DFT)* of $\boldsymbol{x}$ is

$$\mathcal{F}_n : \Pi_n \to \Pi_n \qquad \text{such that} \qquad \mathcal{F}_n \boldsymbol{x} = \boldsymbol{y} \quad \text{where} \quad y_k = \frac{1}{n} \sum_{l=0}^{n-1} \omega_n^{-kl} x_l, \quad k \in \mathbb{Z}.$$

[1]Please email all corrections and suggestions to these notes to `A.Iserles@damtp.cam.ac.uk`. All handouts are available on the WWW at the URL `http://www.damtp.cam.ac.uk/user/na/PartII/Handouts.html`.

Trivial exercise: You can easily prove that $\mathcal{F}_n$ is an isomorphism of Π_n onto itself and that

$$\mathcal{F}_n^{-1}\boldsymbol{y} = \boldsymbol{x}, \qquad \text{where} \qquad x_l = \sum_{k=0}^{n-1} \omega_n^{kl} y_k, \quad l \in \mathbb{Z}.$$

An important observation: Because of (4.8), multiplication by Q can be reduced to calculating an inverse of DFT.
Since we need to evaluate DFT (or its inverse) only in a single period, we can do so by multiplying a vector by a matrix, at the cost of $\mathcal{O}(n^2)$ operations. This, however, is suboptimal and the cost of calculation can be lowered a great deal!

Algorithm 4.19 *The fast Fourier transform (FFT)* Suppose that n is a power of 2, $n = 2^L$ and denote by

$$\boldsymbol{y}^{(\mathrm{E})} = \{y_{2j}\}_{j\in\mathbb{Z}} \qquad \text{and} \qquad \boldsymbol{y}^{(\mathrm{O})} = \{y_{2j+1}\}_{j\in\mathbb{Z}}$$

the even and odd portions of $\boldsymbol{y}$, respectively. Note that $\boldsymbol{y}^{(\mathrm{E})}, \boldsymbol{y}^{(\mathrm{O})} \in \Pi_{n/2}$.
Suppose that we already know the inverse DFT of both 'short' sequences,

$$\boldsymbol{x}^{(\mathrm{E})} = \mathcal{F}_{n/2}^{-1}\boldsymbol{y}^{(\mathrm{E})}, \qquad \boldsymbol{x}^{(\mathrm{O})} = \mathcal{F}_{n/2}^{-1}\boldsymbol{y}^{(\mathrm{O})}.$$

It is then possible to assemble $\boldsymbol{x} = \mathcal{F}_n^{-1}\boldsymbol{y}$ in a small number of operations. Since $\omega_n^n = 1$, we have

$$\begin{aligned} x_l &= \sum_{j=0}^{2^L-1} \omega_{2^L}^{jl} y_j = \sum_{j=0}^{2^{L-1}-1} \omega_{2^L}^{2jl} y_{2j} + \sum_{j=0}^{2^{L-1}-1} \omega_{2^L}^{(2j+1)l} y_{2j+1} \\ &= \sum_{j=0}^{2^{L-1}-1} \omega_{2^{L-1}}^{jl} y_j^{(\mathrm{E})} + \omega_{2^L}^{l} \sum_{j=0}^{2^{L-1}-1} \omega_{2^{L-1}}^{jl} y_j^{(\mathrm{O})} = x_l^{(\mathrm{E})} + \omega_{2^L}^{l} x_l^{(\mathrm{O})}, \quad l = 0, \dots, 2^L - 1. \end{aligned}$$

Therefore, it costs just n products to evaluate $\boldsymbol{x}$, provided that $\boldsymbol{x}^{(\mathrm{E})}$ and $\boldsymbol{x}^{(\mathrm{O})}$ are known. This, incidentally, can be further reduced to $\frac{1}{2}n$, since for $l = 0, 1, \dots, n/2 - 1 = 2^{L-1} - 1$ we have

$$\omega_{2^L}^{l+2^{L-1}} = \omega_{2^L}^{2^{L-1}} \omega_{2^L}^{l} = -\omega_{2^L}^{l} \qquad \Longrightarrow \qquad x_{l+2^{L-1}} = x_l^{(\mathrm{E})} - \omega_{2^L}^{l} x_l^{(\mathrm{O})}.$$

Thus, the products $\omega_{2^L}^{l} x_l^{(\mathrm{O})}$ need be evaluated only for $l \le n/2 - 1$.

To execute FFT, we start from vectors of unit length and in each sth stage, $s = 1, 2, \dots, L$, assemble 2^{L-s} vectors of length 2^s from vectors of length 2^{s-1}: this 'costs' $2^{L-1} = \frac{1}{2}n$ products. Altogether, the cost of FFT is $\frac{1}{2}n \log_2 n$ products.

For $n = 1024 = 2^{10}$, say, the cost is $\approx 5\times 10^3$ products, compared to $\approx 10^6$ for naive matrix multiplication! For $n = 2^{20}$ the respective numbers are $\approx 1.05 \times 10^7$ and $\approx 1.1 \times 10^{12}$, which represents a saving by a factor of more than 10^5.

Applications 4.20 The FFT has numerous further applications: spectral and pseudospectral discretization methods for PDEs, computation of Fourier harmonics, image processing, filtering and reconstruction, computer vision, solution of integral equations, fast multiplication of long integers, numerical conformal maps, ... Arguably, it is the most valuable and widely-used computational technique.
Perhaps the most useful is the computation of *Fourier coefficients*

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{e}^{-\mathrm{i}kx} f(x)\mathrm{d}x \approx \frac{1}{n}\sum_{m=0}^{n-1} \omega_n^{-km} f(\tfrac{2\pi m}{n}), \qquad -\lfloor n/2\rfloor + 1 \le k \le \lfloor n/2\rfloor.$$

If f is periodic and analytic in the strip $\{z \in \mathbb{C} : -\pi \le \operatorname{Re} z \le \pi,\ |\operatorname{Im} z| \le \alpha\}$ then the error decays like $\mathcal{O}(\mathrm{e}^{-\alpha n})$.
The FFT was discovered by Gauss (and forgotten), rediscovered by Lanczos (and forgotten) and, finally, rediscovered by Cooley and Tuckey (and changed history).