$\ell = 1, 2, \ldots, m_2$ (recall that $Q = Q^{-1}$).

Let us review briefly the stages in this, the *Hockney method*, estimating their computational cost.

(1) To start with, we have the vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{m_2} \in \mathbb{R}^{m_1}$ and we form the products $\boldsymbol{c}_\ell = Q\boldsymbol{b}_\ell$, $\ell = 1, 2, \ldots, m_2$. This costs $\mathcal{O}(m_1^2 m_2)$ operations.

(2) We rearrange columns into rows, i.e., $\boldsymbol{c}_\ell \in \mathbb{R}^{m_1}$, $\ell = 1, 2, \ldots, m_2$, into $\tilde{\boldsymbol{c}}_j \in \mathbb{R}^{m_2}$, $j = 1, 2, \ldots, m_1$. This is a purely a change of notation and is free of any computational cost.

(3) The tridiagonal systems $\Gamma_j \tilde{\boldsymbol{y}}_j = \tilde{\boldsymbol{c}}_j$, $j = 1, 2, \ldots, m_1$, are solved by banded LU factorization, and the cost of this procedure is $\mathcal{O}(m_1 m_2)$.

(4) We next rearrange rows into columns, i.e., $\tilde{\boldsymbol{y}}_j \in \mathbb{R}^{m_2}$, $j = 1, 2, \ldots, m_1$ into $\boldsymbol{y}_\ell \in \mathbb{R}^{m_1}$, $\ell = 1, 2, \ldots, m_2$. Again, this costs nothing.

(5) Finally, we find the solution of the discretized Poisson equation by forming the products $\boldsymbol{x}_\ell = Q\boldsymbol{y}_\ell$, $\ell = 1, 2, \ldots, m_2$, at the cost of $\mathcal{O}(m_1^2 m_2)$ operations.

Provided both m_1 and m_2 are large, matrix multiplications dominate the computational cost. Our first, trivial observation is that, the expense being $\mathcal{O}(m_1^2 m_2)$, it is a good policy to choose $m_1 \le m_2$ (because of symmetry, we can always rotate the rectangle, in other words proceed from row to columns, and to rows again). However, a considerably more important observation is that the special form of the matrix Q can be exploited to make products of the form $\boldsymbol{s} = Q\boldsymbol{p}$, say, substantially cheaper than $\mathcal{O}(m_1^2)$. Because of (10.29), we have

$$s_\ell = c\sum_{j=1}^{m_1} p_j \sin\left(\frac{\pi j \ell}{m_1+1}\right) = c\,\mathrm{Im}\left[\sum_{j=0}^{m_1} p_j \exp\left(\frac{\pi \mathrm{i} j \ell}{m_1+1}\right)\right], \qquad \ell = 1, 2, \ldots, m_1, \tag{12.6}$$

where $c = [2/(m_1+1)]^{1/2}$ is a multiplicative constant. Such a product can be performed in a very efficient manner by using a fast Fourier transform, the theme of the next section.

An important remark is that in this section we have not used the fact that we are solving the Poisson equation! The crucial feature of the underlying linear system is that it is block-TST and each of the blocks is itself a TST matrix. There is nothing to prevent us from using the same approach for other equations that possess this structure, regardless of their origin. Moreover, it is an easy matter to extend this approach to, say, Poisson equations in three dimensions with Dirichlet conditions along the boundary of a parallelepiped: the matrix partitions into blocks of block-TST matrices and each such block-TST matrix is itself composed of TST matrices.

12.2 The fast Fourier transform

Let d be a positive integer and denote by Π_d the set of all complex sequences $\boldsymbol{x} = \{x_j\}_{j=-\infty}^{\infty}$ which are periodic with period d, i.e., which are such that $x_{j+d} = x_j$, $j \in \mathbb{Z}$.

It is an easy matter to demonstrate that Π_d is a linear space of dimension d over the complex numbers $\mathbb{C}$ (see Exercise 12.3).

Let

$$\omega_d := \exp\left(\frac{2\pi \mathrm{i}}{d}\right)$$

denote the *primitive root of unity* of degree d. A *discrete Fourier transform (DFT)* is a linear mapping $\mathcal{F}_d$ defined for every $\boldsymbol{x} \in \Pi_d$ by

$$\boldsymbol{y} = \mathcal{F}_d \boldsymbol{x} \qquad \text{where} \qquad y_j = \frac{1}{d}\sum_{\ell=0}^{d-1} \omega_d^{-j\ell} x_\ell, \quad j \in \mathbb{Z}. \tag{12.7}$$

Lemma 12.1 *The DFT $\mathcal{F}_d$, as defined in* (12.7), *maps Π_d into itself. The mapping is invertible and*

$$\boldsymbol{x} = \mathcal{F}_d^{-1}\boldsymbol{y} \qquad \textit{where} \qquad x_\ell = \sum_{j=0}^{d-1} \omega_d^{j\ell} y_j, \quad \ell \in \mathbb{Z}. \tag{12.8}$$

Moreover, $\mathcal{F}_d$ is an isomorphism of Π_d onto itself (A.1.4.2 and A.2.1.9).

Proof Since ω_d is a root of unity, it is true that $\omega_d^d = \omega_d^{-d} = 1$. Therefore it follows from (12.7) that

$$y_{j+d} = \frac{1}{d}\sum_{\ell=0}^{d-1} \omega_d^{-(j+d)\ell} x_\ell = \frac{1}{d}\sum_{\ell=0}^{d-1} \omega_d^{-j\ell} x_\ell = y_j, \qquad j \in \mathbb{Z},$$

and we deduce that $\boldsymbol{y} \in \Pi_d$. Therefore $\mathcal{F}_d$ indeed maps elements of Π_d into elements of Π_d.

To prove the stipulated form of the inverse, we denote $\boldsymbol{w} := \mathcal{F}_d\boldsymbol{x}$, where $\boldsymbol{x}$ has been defined in (12.8). Our first observation is that, provided $\boldsymbol{y} \in \Pi_d$, it is also true that $\boldsymbol{x} \in \Pi_d$ (just swap minus to plus in the above proof). Moreover, changing the order of summation,

$$\begin{aligned} w_m &= \frac{1}{d}\sum_{\ell=0}^{d-1} \omega_d^{-m\ell} x_\ell = \frac{1}{d}\sum_{\ell=0}^{d-1} \omega_d^{-m\ell}\left(\sum_{j=0}^{d-1} \omega_d^{j\ell} y_j\right) \\ &= \frac{1}{d}\sum_{j=0}^{d-1}\left(\sum_{\ell=0}^{d-1} \omega_d^{(j-m)\ell}\right) y_j, \qquad m \in \mathbb{Z}. \end{aligned}$$

Within the parentheses is a geometric series that can be summed explicitly. If $j \neq m$, we have

$$\sum_{\ell=0}^{d-1} \omega_d^{(j-m)\ell} = \frac{1 - \omega_d^{(j-m)d}}{1 - \omega_d^{j-m}} = 0,$$

because $\omega_d^{sd} = 1$ for every $s \in \mathbb{Z} \setminus \{0\}$; whereas in the case $j = m$ we obtain

$$\sum_{\ell=0}^{d-1} \omega_d^{(j-m)d} = \sum_{\ell=0}^{d-1} 1 = d.$$

We thus conclude that $w_m = y_m$, $m \in \mathbb{Z}$, hence that $\boldsymbol{w} = \boldsymbol{y}$. Therefore, (12.8) indeed describes the inverse of $\mathcal{F}_d$.

The existence of an inverse for every $\boldsymbol{x} \in \Pi_d$ shows that $\mathcal{F}_d$ is an isomorphism. To conclude the proof and demonstrate that this DFT maps Π_d onto itself, we suppose that there exists a $\tilde{\boldsymbol{y}} \in \Pi_d$ that cannot be the destination of $\mathcal{F}_d\boldsymbol{x}$ for any $\boldsymbol{x} \in \Pi_d$. Let us define $\tilde{\boldsymbol{x}}$ in terms of (12.8). Then, as we have already observed, $\tilde{\boldsymbol{x}} \in \Pi_d$ and it follows from our proof that $\tilde{\boldsymbol{y}} = \mathcal{F}_d\tilde{\boldsymbol{x}}$. Therefore $\tilde{\boldsymbol{y}}$ is in the range of $\mathcal{F}_d$, in contradiction to our assumption, and we conclude that the DFT $\mathcal{F}_d$ is, indeed, an isomorphism of Π_d *onto* itself. ■

It is of interest to mention an alternative proof that $\mathcal{F}_d$ is onto. According to a classical theorem from linear algebra, a linear mapping T from a finite-dimensional linear space V to itself is onto if and only if its kernel $\ker T$ consists just of the zero element of the space ($\ker T$ is the set of all $\boldsymbol{w} \in V$ such that $T\boldsymbol{w} = \boldsymbol{0}$; see A.1.4.2). Letting $\boldsymbol{x} \in \ker \mathcal{F}_d$, (12.7) yields

$$\sum_{\ell=0}^{d-1} \omega_d^{-j\ell} x_\ell = 0, \qquad j = 0, 1, \ldots, d-1. \tag{12.9}$$

This is a homogeneous linear system of d equations in the d unknowns $x_0, x_1, \ldots, x_{d-1}$. Its matrix is a *Vandermonde* matrix (A.1.2.5) and it is easy to prove that its determinant satisfies

$$\det \begin{bmatrix} 1 & \omega_d^{-1} & \cdots & \omega_d^{-(d-1)} \\ 1 & \omega_d^{-2} & \cdots & \omega_d^{-2(d-1)} \\ \vdots & \vdots & & \vdots \\ 1 & \omega_d^{-d} & \cdots & \omega_d^{-d(d-1)} \end{bmatrix} = \prod_{\ell=1}^{d} \prod_{j=0}^{\ell-1} (\omega_d^{-\ell} - \omega_d^{-j}) \neq 0.$$

Therefore the only possible solution of (12.9) is $x_0 = x_1 = \cdots = x_{d-1} = 0$ and we deduce that $\ker \mathcal{F}_d = \boldsymbol{0}$ and the mapping is onto Π_d.

◇ **Applications of the DFT** It is difficult to overstate the importance of the discrete Fourier transform to a multitude of applications, ranging from numerical analysis to control theory, and from computer science to coding theory to time series analysis.... In the sequel we employ it to provide a fast solution to discretized differential equations but here we give another example that shows the power of the DFT: approximation of the *Fourier coefficients,* also known as *Fourier harmonics,* of a periodic function. This theme finds its application in the next section, as well as in numerous other places across applied mathematics, science and engineering.

Let g be an integrable complex-valued function, defined for all $x \in \mathbb{R}$, and periodic with period 2π (i.e., $g(x + 2\pi) = g(x)$ for all $x \in \mathbb{R}$). The *Fourier transform* of g is the sequence $\{\hat{g}_m\}_{m=-\infty}^{\infty}$, where

$$\hat{g}_m = \frac{1}{2\pi} \int_0^{2\pi} g(\tau) \mathrm{e}^{-\mathrm{i}m\tau} \, \mathrm{d}\tau, \qquad m \in \mathbb{Z}. \tag{12.10}$$

Fourier transforms feature in numerous branches of mathematical analysis and its application: the library shelf labelled 'harmonic analysis' is, to a very large extent, devoted to Fourier transforms and their ramifications. More to the point, as far as the subject matter of this book is concerned, they are crucial in the stability analysis of numerical methods for PDEs of evolution (see Chapters 13 and 14).

The ubiquity of Fourier transforms in so many applications means that the task of approximating (12.10) numerically is among the most important problems in scientific computing. The obvious approach is to pursue the logic of Section 3.1 and replace integration by finite summation, i.e., by quadrature. It is not difficult to prove that, provided $g(0) = g(2\pi)$, the best quadrature nodes – our equivalent of the Gaussian quadrature nodes of Section 3.1 – are equidistant along the interval $[0, 2\pi]$, and furthermore the corresponding quadrature weights are all equal. In other words, given an integer $\nu \geq 1$, we approximate

$$\hat{g}_m \approx \hat{h}_m := \frac{1}{\nu}\sum_{j=0}^{\nu-1} g\left(\frac{2\pi j}{\nu}\right) \exp\left(\frac{-2\pi \mathrm{i} m j}{\nu}\right) = \frac{1}{\nu}\sum_{j=0}^{\nu-1} \omega_\nu^{-mj} g_j, \qquad (12.11)$$

where $g_j = g(2\pi j/\nu)$, $j = 0, 1, \ldots, \nu-1$. In a rare display of good mathematical fortune, virtue – Gaussian nodes – combines with ease of application: equidistant nodes and equal weights.

Of course, (12.11) makes sense only for a finite set of indices m and we are safe only if we restrict this to just ν values. Suppose for simplicity that ν is even and let m vary in $\{-\nu/2+1, -\nu/2+2, \ldots, \nu/2\}$. Replacing m by $\ell - \nu/2 + 1$, the identity $\omega_\nu^{\nu/2} = -1$ gives

$$\begin{aligned}\hat{h}_m &= \frac{1}{\nu}\sum_{j=0}^{\nu-1} \omega_\nu^{-(\ell-\nu/2+1)j} g_j = \left(\frac{\omega_\nu^{\nu/2-1}}{\nu}\right)\sum_{j=0}^{\nu-1} \omega_\nu^{-\ell j} g_j \\ &= -\,\omega_\nu^{-1}\,(\mathcal{F}_\nu \boldsymbol{g})_\ell, \qquad \ell = 0, 1, \ldots, \nu-1,\end{aligned}$$

where $\boldsymbol{g} = \{g_j\}_{j=-\infty}^{\infty} \in \Pi_\nu$ is obtained by letting $g_{j+s\nu} = g_j$ for every $j = 0, 1, \ldots, \nu-1$ and integer s. In other words, (12.11) is nothing other but a DFT!

Suppose that the *Fourier series* corresponding to g is absolutely convergent, hence

$$g(x) = \sum_{k=-\infty}^{\infty} \hat{g}_k \mathrm{e}^{\mathrm{i}mx}, \qquad 0 \leq x \leq 2\pi. \qquad (12.12)$$

(This, incidentally, is the formula that reverses the action of the Fourier transform. Its representation of g as a linear combination of harmonics is perhaps the most important reason for the popularity of Fourier series – and Fourier transforms – in applications.) Letting $x = 2\pi j/\nu$, we obtain

$$g_j = \sum_{k=-\infty}^{\infty} \hat{g}_k \exp\left(\frac{2\pi jk}{\nu}\right), \qquad j = 0, 1, \ldots, \nu-1,$$

and substitution into (12.11) yields

$$\hat{h}_m = \frac{1}{\nu}\sum_{j=0}^{\nu-1}\omega_\nu^{-mj}\left(\sum_{k=-\infty}^{\infty}\hat{g}_k\omega_\nu^{jk}\right) = \frac{1}{\nu}\sum_{k=-\infty}^{\infty}\hat{g}_k\sum_{j=0}^{\nu-1}\omega_\nu^{(k-m)j}.$$

We are justified in replacing the order of summation since we have assumed that the Fourier series converges absolutely. However, as we have already seen,

$$\sum_{j=0}^{\nu-1}\omega_\nu^{jr} = \begin{cases} 0, & r \bmod \nu \neq 0, \\ \nu, & r \bmod \nu = 0 \end{cases}$$

and we deduce that the error in our approximation of the Fourier coefficients $\hat{g}_m$ is

$$\hat{h}_m - \hat{g}_m = \sum_{\substack{j=-\infty \\ j\neq 0}}^{\infty}\hat{g}_{m+j\nu}. \tag{12.13}$$

Therefore, we can expect the error to decrease rapidly as a function of ν if the Fourier coefficients of g decay fast for large $|k|$.

The rate of decay of Fourier coefficients is known for a large variety of functions. In particular, suppose that (12.12) converges for all $z \in \mathbb{C}$ such that $0 \le \operatorname{Re} z \le 2\pi$, $|\operatorname{Im} z| \le \alpha$ for some $\alpha > 0$. Then it is possible to show that

$$|\hat{g}_k| \le c\,\mathrm{e}^{-|k|\alpha}, \qquad k \in \mathbb{Z},$$

where $c > 0$ is a constant. Therefore, by dominating (12.13) with a geometric series we can deduce that

$$|\hat{h}_m - \hat{g}_m| \le 2c\left(\frac{\mathrm{e}^{-\nu\alpha}}{1-\mathrm{e}^{-\nu\alpha}}\right)\cosh m\alpha, \qquad |m| \le \nu - 1.$$

Therefore, the error in replacing the Fourier integral by a DFT decays *exponentially* with ν. $\diamond$

On the face of it, the evaluation of the DFT (12.7) (or of its inverse) requires $\mathcal{O}(d^2)$ operations, since, by periodicity, it is obtained by multiplying a vector in $\mathbb{C}^d$ by a $d \times d$ complex matrix. It is one of the great wonders of computational mathematics, however, that this operation count can be reduced a very great deal. Let us assume for simplicity that $d = 2^n$, where n is a nonnegative integer. It is convenient to replace $\mathcal{F}_d$ by the mapping $\mathcal{F}_d^* := d\mathcal{F}_d$; clearly, if we can compute $\mathcal{F}_d^*\boldsymbol{x}$ cheaply, just $\mathcal{O}(d)$ operations will convert the result to $\mathcal{F}_d\boldsymbol{x}$.

Let us define, for every $\boldsymbol{x} \in \Pi_d$,

$$\boldsymbol{x}^{[\mathrm{e}]} := \{x_{2j}\}_{j=-\infty}^{\infty} \qquad \text{and} \qquad \boldsymbol{x}^{[\mathrm{o}]} := \{x_{2j+1}\}_{j=-\infty}^{\infty}.$$

Since $\boldsymbol{x}^{[\mathrm{e}]}, \boldsymbol{x}^{[\mathrm{o}]} \in \Pi_{d/2}$, we can make the mapppings

$$\boldsymbol{y}^{[\mathrm{e}]} = \mathcal{F}_{d/2}^*\boldsymbol{x}^{[\mathrm{e}]} \qquad \text{and} \qquad \boldsymbol{y}^{[\mathrm{o}]} = \mathcal{F}_{d/2}^*\boldsymbol{x}^{[\mathrm{o}]}.$$

Let $\boldsymbol{y} = \mathcal{F}_d^* \boldsymbol{x}$. Then, by (12.7),

$$\begin{aligned} y_j &= \sum_{\ell=0}^{d-1} \omega_d^{-j\ell} x_\ell = \sum_{\ell=0}^{2^n-1} \omega_{2^n}^{-j\ell} x_\ell \\ &= \sum_{\ell=0}^{2^{n-1}-1} \omega_{2^n}^{-2j\ell} x_{2\ell} + \sum_{\ell=0}^{2^{n-1}-1} \omega_{2^n}^{-j(2\ell+1)} x_{2\ell+1}, \qquad j = 0, 1, \ldots, 2^n - 1. \end{aligned}$$

However,

$$\omega_{2^n}^{2s} = \omega_{2^{n-1}}^{s}, \qquad s \in \mathbb{Z},$$

therefore

$$\begin{aligned} y_j &= \sum_{j=0}^{2^{n-1}-1} \omega_{2^{n-1}}^{-j\ell} x_{2\ell} + \omega_{2^n}^{-j} \sum_{j=0}^{2^{n-1}-1} \omega_{2^{n-1}}^{-j\ell} x_{2\ell+1} \\ &= y_j^{[\mathrm{e}]} + \omega_{2^n}^{-j} y_j^{[\mathrm{o}]}, \qquad j = 0, 1, \ldots, 2^n - 1. \end{aligned} \tag{12.14}$$

In other words, provided that $\boldsymbol{y}^{[\mathrm{e}]}$ and $\boldsymbol{y}^{[\mathrm{o}]}$ are already known, we can synthesize them into $\boldsymbol{y}$ in $\mathcal{O}(d)$ operations.

Incidentally – and before proceeding any further – we observe that the number of operations can be significantly reduced by exploiting the identity $\omega_{2^s}^{-s} = -1$, $s \geq 1$. Hence, (12.14) yields

$$\begin{aligned} y_j &= y_j^{[\mathrm{e}]} + \omega_{2^n}^{-j} y_j^{[\mathrm{o}]}, \\ y_{j+2^{n-1}} &= y_{j+2^{n-1}}^{[\mathrm{e}]} + \omega_{2^n}^{-j-2^{n-1}} y_{j+2^{n-1}}^{[\mathrm{o}]} = y_j^{[\mathrm{e}]} - \omega_{2^n}^{-j} y_j^{[\mathrm{o}]}, \end{aligned} \qquad j = 0, 1, \ldots, 2^{n-1}-1$$

(recall that $\boldsymbol{y}^{[\mathrm{e}]}, \boldsymbol{y}^{[\mathrm{o}]} \in \Pi_{2^{n-1}}$, therefore $y_{j+2^{n-1}}^{[\mathrm{e}]} = y_j^{[\mathrm{e}]}$ etc.). In other words, to combine $\boldsymbol{y}^{[\mathrm{e}]}$ and $\boldsymbol{y}^{[\mathrm{o}]}$, we need to form just 2^{n-1} products $\omega_{2^{n-1}}^{-j} y_j^{[\mathrm{o}]}$, subsequently adding or subtracting them, as required, from $y_j^{[\mathrm{e}]}$ for $j = 0, 1, \ldots, 2^{n-1}$.

All this, needless to say, is based on the premise that $\boldsymbol{y}^{[\mathrm{e}]}$ and $\boldsymbol{y}^{[\mathrm{o}]}$ are known, which, as things stand, is false. Having said this, we can form, in a similar fashion to that above, $\boldsymbol{y}^{[\mathrm{e}]}$ from $\mathcal{F}_{d/4}^* \boldsymbol{x}^{[\mathrm{ee}]}, \mathcal{F}_{d/4}^* \boldsymbol{x}^{[\mathrm{eo}]} \in \Pi_{d/4}$, where

$$\boldsymbol{x}^{[\mathrm{ee}]} = \{x_{4j}\}_{j=-\infty}^{\infty}, \qquad \boldsymbol{x}^{[\mathrm{eo}]} = \{x_{4j+2}\}_{j=-\infty}^{\infty}.$$

Likewise, $\boldsymbol{y}^{[\mathrm{o}]}$ can be also obtained from two transforms of length $d/4$. This procedure can be iterated until we reach transforms of unit length, which, of course, are the variables themselves.

Practical implementation of this procedure, the famous *fast Fourier transform* (FFT), proceeds from the other end: we commence from 2^n transforms of length 1 and synthesize them to 2^{n-1} transforms of length 2. These are, in turn, combined into 2^{n-2} transforms of length 2^2, next into 2^{n-3} transforms of length 2^3 and so on, until we reach a single transform of length 2^n, the object of this whole exercise.

Assembling 2^{n-s+1} transforms of length 2^{s-1} into 2^{n-s} transforms of double the length costs $\mathcal{O}(2^{n-s} \times 2^s) = \mathcal{O}(d)$ operations. Since there are n such 'layers' the total

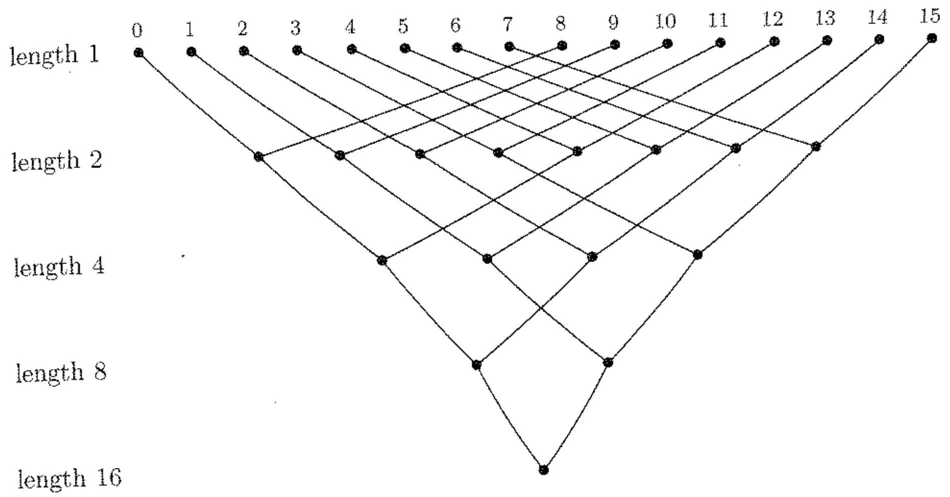

Figure 12.1 The assembly pattern in FFT for $d = 16 = 2^4$.

expense of the FFT is a multiple of $2^n n = d \log_2 d$ operations. For large values of d this results in a very significant saving, compared with naive matrix multiplication.

The order of assembly of one set of transforms into new transforms of twice the length is important – we do not just combine any two arbitrary 'strands'! The correct arrangement is displayed in Figure 12.1 in the case $n = 4$ and the general rule is obvious. It can be best understood by expressing the index ℓ in a binary representation, but we choose not to dwell further on that issue.

Having introduced FFTs, we use them to decrease the computational cost of the Hockney method from Section 12.1. Recall that the problematic part of the calculation consists of $2m_2$ products of the form (12.6). We recognize readily each such product as an inverse DFT in disguise,

$$s_\ell = c \operatorname{Im} \left(\sum_{j=0}^{m_1} p_j \omega_{2m_1+2}^{j\ell} \right), \qquad \ell = 1, 2, \ldots, m_1.$$

To bring this into a proper DFT form, we set $p_j = 0$, $j = m_1+1, m_1+2, \ldots, 2m_1+1$ and extend the range of ℓ to $\{0, 1, \ldots, 2m_1+1\}$. This yields

$$s_\ell = c \operatorname{Im} \left(\sum_{j=0}^{2m_1+1} p_j \omega_{2m_1+2}^{j\ell} \right), \qquad \ell = 0, 1, \ldots, 2m_1+1,$$

that is

$$\tilde{\boldsymbol{s}} = c \operatorname{Im} \mathcal{F}_{2m_1+1}^{-1} \tilde{\boldsymbol{p}},$$

where $\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{s}} \in \Pi_{2m_1+2}$. Finally, we discard s_0 and $s_{m_1+1}, s_{m_1+2}, \ldots, s_{2m_1+1}$ and retain the remaining values as the result of our calculation.

Even when we calculate a system twice the size and employ complex arithmetic, this procedure results in substantial savings even for moderate values of m_1.[1]

The fast Fourier transform is instrumental to the performance of most fast Poisson solvers, and we return to this theme in the next section, in the context of fast solution of the Poisson equation in a disc. There are exceptions and, in the comments following this chapter, we briefly mention the method of *cyclic odd–even reduction and factorization,* which requires no FFT.

This is not the only application of FFT to the numerical solution of PDEs – indeed, it is hardly the most important. This distinction, in fairness, belongs to spectral methods, whose very power rests on the availability of FFTs and other 'fast transforms'.

12.3 Fast Poisson solver in a disc

Let us suppose that the Poisson equation $\nabla^2 u = g_0$ is given in the open unit disc

$$\mathbb{D} = \{(x,y) \in \mathbb{R}^2 \,:\, x^2 + y^2 < 1\},$$

together with Dirichlet boundary conditions along the unit circle,

$$u(\cos\theta, \sin\theta) = \phi(\theta), \qquad 0 \le \theta \le 2\pi, \tag{12.15}$$

where $\phi(0) = \phi(2\pi)$. It is convenient to translate the equation from Cartesian to polar coordinates. Thus, we let

$$\begin{aligned} v(r,\theta) &= u(r\cos\theta, r\sin\theta), \\ g(r,\theta) &= g_0(r\cos\theta, r\sin\theta), \end{aligned} \qquad 0 < r < 1, \quad 0 \le \theta \le 2\pi.$$

The form of ∇^2 in polar coordinates readily gives us

$$\frac{\partial^2 v}{\partial r^2} + \frac{1}{r}\frac{\partial v}{\partial r} + \frac{1}{r^2}\frac{\partial^2 v}{\partial \theta^2} = g, \qquad 0 < r < 1, \quad 0 \le \theta \le 2\pi. \tag{12.16}$$

The boundary conditions, however, are more delicate. Switching from Cartesian to polar means, in essence, that the disc $\mathbb{D}$ is replaced by the square

$$\widetilde{\mathbb{D}} = \{(r,\theta) \,:\, 0 < r < 1,\ 0 \le \theta \le 2\pi\}.$$

Unlike $\mathbb{D}$, which has just one boundary 'segment' – its circumference – the set $\widetilde{\mathbb{D}}$ boasts four portions of boundary and we need to allocate appropriate conditions at all of them.

The segment $r = 1$, $0 \le \theta \le 2\pi$, is the easiest, being the destination of the original boundary condition (12.15). Hence, we set

$$v(1,\theta) = \phi(\theta), \qquad 0 \le \theta \le 2\pi. \tag{12.17}$$

Next in order of difficulty are the line segments $0 < r < 1$, $\theta = 0$, and $0 < r < 1$, $\theta = 2\pi$. They both correspond to the same segment, namely $0 < x < 1$, $y = 0$, in the

[1]As a matter of fact, the calculation can be performed in real arithmetic, using the so-called *fast sine transform.*