For example, a combined advective-diffusion equation, such as

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} + D\frac{\partial^2 u}{\partial x^2} \tag{19.3.21}$$

might profitably use an explicit scheme for the advective term combined with a Crank-Nicholson or other implicit scheme for the diffusion term.

The alternating-direction implicit (ADI) method, equation (19.3.16), is an example of operator splitting with a slightly different twist. Let us reinterpret (19.3.19) to have a different meaning: Let $\mathcal{U}_1$ now denote an updating method that includes algebraically *all* the pieces of the total operator $\mathcal{L}$, but which is desirably *stable* only for the $\mathcal{L}_1$ piece; likewise $\mathcal{U}_2, \dots \mathcal{U}_m$. Then a method of getting from u^n to u^{n+1} is

$$\begin{aligned} u^{n+1/m} &= \mathcal{U}_1(u^n, \Delta t/m) \\ u^{n+2/m} &= \mathcal{U}_2(u^{n+1/m}, \Delta t/m) \\ &\cdots \\ u^{n+1} &= \mathcal{U}_m(u^{n+(m-1)/m}, \Delta t/m) \end{aligned} \tag{19.3.22}$$

The timestep for each fractional step in (19.3.22) is now only $1/m$ of the full timestep, because each partial operation acts with all the terms of the original operator.

Equation (19.3.22) is usually, though not always, stable as a differencing scheme for the operator $\mathcal{L}$. In fact, as a rule of thumb, it is often sufficient to have stable $\mathcal{U}_i$'s only for the operator pieces having the highest number of spatial derivatives — the other $\mathcal{U}_i$'s can be *unstable* — to make the overall scheme stable!

It is at this point that we turn our attention from initial value problems to boundary value problems. These will occupy us for the remainder of the chapter.

CITED REFERENCES AND FURTHER READING:

Ames, W.F. 1977, *Numerical Methods for Partial Differential Equations*, 2nd ed. (New York: Academic Press).

19.4 Fourier and Cyclic Reduction Methods for Boundary Value Problems

As discussed in §19.0, most boundary value problems (elliptic equations, for example) reduce to solving large sparse linear systems of the form

$$\mathbf{A} \cdot \mathbf{u} = \mathbf{b} \tag{19.4.1}$$

either once, for boundary value equations that are linear, or iteratively, for boundary value equations that are nonlinear.

Two important techniques lead to "rapid" solution of equation (19.4.1) when the sparse matrix is of certain frequently occurring forms. The *Fourier transform method* is directly applicable when the equations have coefficients that are constant in space. The *cyclic reduction* method is somewhat more general; its applicability is related to the question of whether the equations are separable (in the sense of "separation of variables"). Both methods require the boundaries to coincide with the coordinate lines. Finally, for some problems, there is a powerful combination of these two methods called *FACR (Fourier Analysis and Cyclic Reduction)*. We now consider each method in turn, using equation (19.0.3), with finite-difference representation (19.0.6), as a model example. Generally speaking, the methods in this section are faster, when they apply, than the simpler relaxation methods discussed in §19.5; but they are not necessarily faster than the more complicated multigrid methods discussed in §19.6.

Fourier Transform Method

The discrete inverse Fourier transform in both x and y is

$$u_{jl} = \frac{1}{JL} \sum_{m=0}^{J-1} \sum_{n=0}^{L-1} \widehat{u}_{mn} e^{-2\pi ijm/J} e^{-2\pi iln/L} \tag{19.4.2}$$

This can be computed using the FFT independently in each dimension, or else all at once via the routine `fourn` of §12.4 or the routine `rlft3` of §12.5. Similarly,

$$\rho_{jl} = \frac{1}{JL} \sum_{m=0}^{J-1} \sum_{n=0}^{L-1} \widehat{\rho}_{mn} e^{-2\pi ijm/J} e^{-2\pi iln/L} \tag{19.4.3}$$

If we substitute expressions (19.4.2) and (19.4.3) in our model problem (19.0.6), we find

$$\widehat{u}_{mn} \left(e^{2\pi im/J} + e^{-2\pi im/J} + e^{2\pi in/L} + e^{-2\pi in/L} - 4 \right) = \widehat{\rho}_{mn} \Delta^2 \tag{19.4.4}$$

or

$$\widehat{u}_{mn} = \frac{\widehat{\rho}_{mn} \Delta^2}{2 \left(\cos \dfrac{2\pi m}{J} + \cos \dfrac{2\pi n}{L} - 2 \right)} \tag{19.4.5}$$

Thus the strategy for solving equation (19.0.6) by FFT techniques is:

- Compute $\widehat{\rho}_{mn}$ as the Fourier transform

$$\widehat{\rho}_{mn} = \sum_{j=0}^{J-1} \sum_{l=0}^{L-1} \rho_{jl}\, e^{2\pi imj/J} e^{2\pi inl/L} \tag{19.4.6}$$

- Compute $\widehat{u}_{mn}$ from equation (19.4.5).

- Compute u_{jl} by the inverse Fourier transform (19.4.2).

The above procedure is valid for periodic boundary conditions. In other words, the solution satisfies

$$u_{jl} = u_{j+J,l} = u_{j,l+L} \tag{19.4.7}$$

Next consider a Dirichlet boundary condition $u = 0$ on the rectangular boundary. Instead of the expansion (19.4.2), we now need an expansion in sine waves:

$$u_{jl} = \frac{2}{J}\frac{2}{L}\sum_{m=1}^{J-1}\sum_{n=1}^{L-1} \widehat{u}_{mn} \sin\frac{\pi jm}{J} \sin\frac{\pi ln}{L} \tag{19.4.8}$$

This satisfies the boundary conditions that $u = 0$ at $j = 0, J$ and at $l = 0, L$. If we substitute this expansion and the analogous one for ρ_{jl} into equation (19.0.6), we find that the solution procedure parallels that for periodic boundary conditions:

- Compute $\widehat{\rho}_{mn}$ by the sine transform

$$\widehat{\rho}_{mn} = \sum_{j=1}^{J-1}\sum_{l=1}^{L-1} \rho_{jl} \sin\frac{\pi jm}{J} \sin\frac{\pi ln}{L} \tag{19.4.9}$$

(A fast sine transform algorithm was given in §12.3.)

- Compute $\widehat{u}_{mn}$ from the expression analogous to (19.4.5),

$$\widehat{u}_{mn} = \frac{\Delta^2 \widehat{\rho}_{mn}}{2\left(\cos\frac{\pi m}{J} + \cos\frac{\pi n}{L} - 2\right)} \tag{19.4.10}$$

- Compute u_{jl} by the inverse sine transform (19.4.8).

If we have inhomogeneous boundary conditions, for example $u = 0$ on all boundaries except $u = f(y)$ on the boundary $x = J\Delta$, we have to add to the above solution a solution u^H of the homogeneous equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{19.4.11}$$

that satisfies the required boundary conditions. In the continuum case, this would be an expression of the form

$$u^H = \sum_n A_n \sinh\frac{n\pi x}{J\Delta} \sin\frac{n\pi y}{L\Delta} \tag{19.4.12}$$

where A_n would be found by requiring that $u = f(y)$ at $x = J\Delta$. In the discrete case, we have

$$u^H_{jl} = \frac{2}{L}\sum_{n=1}^{L-1} A_n \sinh\frac{\pi nj}{J} \sin\frac{\pi nl}{L} \tag{19.4.13}$$

If $f(y = l\Delta) \equiv f_l$, then we get A_n from the inverse formula

$$A_n = \frac{1}{\sinh \pi n} \sum_{l=1}^{L-1} f_l \sin \frac{\pi n l}{L} \tag{19.4.14}$$

The complete solution to the problem is

$$u = u_{jl} + u_{jl}^H \tag{19.4.15}$$

By adding appropriate terms of the form (19.4.12), we can handle inhomogeneous terms on any boundary surface.

A much simpler procedure for handling inhomogeneous terms is to note that whenever boundary terms appear on the left-hand side of (19.0.6), they can be taken over to the right-hand side since they are known. The effective source term is therefore ρ_{jl} plus a contribution from the boundary terms. To implement this idea formally, write the solution as

$$u = u' + u^B \tag{19.4.16}$$

where $u' = 0$ on the boundary, while u^B vanishes everywhere *except* on the boundary. There it takes on the given boundary value. In the above example, the only nonzero values of u^B would be

$$u_{J,l}^B = f_l \tag{19.4.17}$$

The model equation (19.0.3) becomes

$$\nabla^2 u' = -\nabla^2 u^B + \rho \tag{19.4.18}$$

or, in finite-difference form,

$$\begin{aligned} u'_{j+1,l} + u'_{j-1,l} + u'_{j,l+1} + u'_{j,l-1} - 4u'_{j,l} = \\ -\left(u^B_{j+1,l} + u^B_{j-1,l} + u^B_{j,l+1} + u^B_{j,l-1} - 4u^B_{j,l}\right) + \Delta^2 \rho_{j,l} \end{aligned} \tag{19.4.19}$$

All the u^B terms in equation (19.4.19) vanish except when the equation is evaluated at $j = J - 1$, where

$$u'_{J,l} + u'_{J-2,l} + u'_{J-1,l+1} + u'_{J-1,l-1} - 4u'_{J-1,l} = -f_l + \Delta^2 \rho_{J-1,l} \tag{19.4.20}$$

Thus the problem is now equivalent to the case of zero boundary conditions, except that one row of the source term is modified by the replacement

$$\Delta^2 \rho_{J-1,l} \rightarrow \Delta^2 \rho_{J-1,l} - f_l \tag{19.4.21}$$

The case of Neumann boundary conditions $\nabla u = 0$ is handled by the cosine expansion (12.3.17):

$$u_{jl} = \frac{2}{J}\frac{2}{L} \sum_{m=0}^{J}{}'' \sum_{n=0}^{L}{}'' \, \widehat{u}_{mn} \cos \frac{\pi j m}{J} \cos \frac{\pi l n}{L} \tag{19.4.22}$$

Here the double prime notation means that the terms for $m = 0$ and $m = J$ should be multiplied by $\frac{1}{2}$, and similarly for $n = 0$ and $n = L$. Inhomogeneous terms $\nabla u = g$ can be again included by adding a suitable solution of the homogeneous equation, or more simply by taking boundary terms over to the right-hand side. For example, the condition

$$\frac{\partial u}{\partial x} = g(y) \qquad \text{at} \quad x = 0 \tag{19.4.23}$$

becomes

$$\frac{u_{1,l} - u_{-1,l}}{2\Delta} = g_l \tag{19.4.24}$$

where $g_l \equiv g(y = l\Delta)$. Once again we write the solution in the form (19.4.16), where now $\nabla u' = 0$ on the boundary. This time ∇u^B takes on the prescribed value on the boundary, but u^B vanishes everywhere except just *outside* the boundary. Thus equation (19.4.24) gives

$$u^B_{-1,l} = -2\Delta g_l \tag{19.4.25}$$

All the u^B terms in equation (19.4.19) vanish except when $j = 0$:

$$u'_{1,l} + u'_{-1,l} + u'_{0,l+1} + u'_{0,l-1} - 4u'_{0,l} = 2\Delta g_l + \Delta^2 \rho_{0,l} \tag{19.4.26}$$

Thus u' is the solution of a zero-gradient problem, with the source term modified by the replacement

$$\Delta^2 \rho_{0,l} \rightarrow \Delta^2 \rho_{0,l} + 2\Delta g_l \tag{19.4.27}$$

Sometimes Neumann boundary conditions are handled by using a staggered grid, with the u's defined midway between zone boundaries so that first derivatives are centered on the mesh points. You can solve such problems using similar techniques to those described above if you use the alternative form of the cosine transform, equation (12.3.23).

Cyclic Reduction

Evidently the FFT method works only when the original PDE has constant coefficients, and boundaries that coincide with the coordinate lines. An alternative algorithm, which can be used on somewhat more general equations, is called *cyclic reduction (CR)*.

We illustrate cyclic reduction on the equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + b(y)\frac{\partial u}{\partial y} + c(y)u = g(x, y) \tag{19.4.28}$$

This form arises very often in practice from the Helmholtz or Poisson equations in polar, cylindrical, or spherical coordinate systems. More general separable equations are treated in [1].

The finite-difference form of equation (19.4.28) can be written as a set of vector equations

$$\mathbf{u}_{j-1} + \mathbf{T}\cdot\mathbf{u}_j + \mathbf{u}_{j+1} = \mathbf{g}_j\Delta^2 \tag{19.4.29}$$

Here the index j comes from differencing in the x-direction, while the y-differencing (denoted by the index l previously) has been left in vector form. The matrix $\mathbf{T}$ has the form

$$\mathbf{T} = \mathbf{B} - 2\mathbf{1} \tag{19.4.30}$$

where the $2\mathbf{1}$ comes from the x-differencing and the matrix $\mathbf{B}$ from the y-differencing. The matrix $\mathbf{B}$, and hence $\mathbf{T}$, is tridiagonal with variable coefficients.

The CR method is derived by writing down three successive equations like (19.4.29):

$$\begin{aligned}
\mathbf{u}_{j-2} + \mathbf{T}\cdot\mathbf{u}_{j-1} + \mathbf{u}_j &= \mathbf{g}_{j-1}\Delta^2\\
\mathbf{u}_{j-1} + \mathbf{T}\cdot\mathbf{u}_j + \mathbf{u}_{j+1} &= \mathbf{g}_j\Delta^2\\
\mathbf{u}_j + \mathbf{T}\cdot\mathbf{u}_{j+1} + \mathbf{u}_{j+2} &= \mathbf{g}_{j+1}\Delta^2
\end{aligned} \tag{19.4.31}$$

Matrix-multiplying the middle equation by $-\mathbf{T}$ and then adding the three equations, we get

$$\mathbf{u}_{j-2} + \mathbf{T}^{(1)}\cdot\mathbf{u}_j + \mathbf{u}_{j+2} = \mathbf{g}_j^{(1)}\Delta^2 \tag{19.4.32}$$

This is an equation of the same form as (19.4.29), with

$$\begin{aligned}
\mathbf{T}^{(1)} &= 2\mathbf{1} - \mathbf{T}^2\\
\mathbf{g}_j^{(1)} &= \Delta^2(\mathbf{g}_{j-1} - \mathbf{T}\cdot\mathbf{g}_j + \mathbf{g}_{j+1})
\end{aligned} \tag{19.4.33}$$

After one level of CR, we have reduced the number of equations by a factor of two. Since the resulting equations are of the same form as the original equation, we can repeat the process. Taking the number of mesh points to be a power of 2 for simplicity, we finally end up with a single equation for the central line of variables:

$$\mathbf{T}^{(f)}\cdot\mathbf{u}_{J/2} = \Delta^2\mathbf{g}_{J/2}^{(f)} - \mathbf{u}_0 - \mathbf{u}_J \tag{19.4.34}$$

Here we have moved $\mathbf{u}_0$ and $\mathbf{u}_J$ to the right-hand side because they are known boundary values. Equation (19.4.34) can be solved for $\mathbf{u}_{J/2}$ by the standard tridiagonal algorithm. The two equations at level $f-1$ involve $\mathbf{u}_{J/4}$ and $\mathbf{u}_{3J/4}$. The equation for $\mathbf{u}_{J/4}$ involves $\mathbf{u}_0$ and $\mathbf{u}_{J/2}$, both of which are known, and hence can be solved by the usual tridiagonal routine. A similar result holds true at every stage, so we end up solving $J-1$ tridiagonal systems.

In practice, equations (19.4.33) should be rewritten to avoid numerical instability. For these and other practical details, refer to [2].

FACR Method

The *best* way to solve equations of the form (19.4.28), including the constant coefficient problem (19.0.3), is a combination of Fourier analysis and cyclic reduction, the FACR method [3-6]. If at the rth stage of CR we Fourier analyze the equations of the form (19.4.32) along y, that is, with respect to the suppressed vector index, we will have a tridiagonal system in the x-direction for each y-Fourier mode:

$$\widehat{u}_{j-2^r}^k + \lambda_k^{(r)}\widehat{u}_j^k + \widehat{u}_{j+2^r}^k = \Delta^2 g_j^{(r)k} \tag{19.4.35}$$

Here $\lambda_k^{(r)}$ is the eigenvalue of $\mathbf{T}^{(r)}$ corresponding to the kth Fourier mode. For the equation (19.0.3), equation (19.4.5) shows that $\lambda_k^{(r)}$ will involve terms like $\cos(2\pi k/L) - 2$ raised to a power. Solve the tridiagonal systems for $\widehat{u}_j^k$ at the levels $j = 2^r, 2 \times 2^r, 4 \times 2^r, ..., J - 2^r$. Fourier synthesize to get the y-values on these x-lines. Then fill in the intermediate x-lines as in the original CR algorithm.

The trick is to choose the number of levels of CR so as to minimize the total number of arithmetic operations. One can show that for a typical case of a 128×128 mesh, the optimal level is $r = 2$; asymptotically, $r \to \log_2(\log_2 J)$.

A rough estimate of running times for these algorithms for equation (19.0.3) is as follows: The FFT method (in both x and y) and the CR method are roughly comparable. FACR with $r = 0$ (that is, FFT in one dimension and solve the tridiagonal equations by the usual algorithm in the other dimension) gives about a factor of two gain in speed. The optimal FACR with $r = 2$ gives another factor of two gain in speed.

CITED REFERENCES AND FURTHER READING:

Swartzrauber, P.N. 1977, *SIAM Review*, vol. 19, pp. 490–501. [1]

Buzbee, B.L, Golub, G.H., and Nielson, C.W. 1970, *SIAM Journal on Numerical Analysis*, vol. 7, pp. 627–656; see also *op. cit.* vol. 11, pp. 753–763. [2]

Hockney, R.W. 1965, *Journal of the Association for Computing Machinery*, vol. 12, pp. 95–113. [3]

Hockney, R.W. 1970, in *Methods of Computational Physics*, vol. 9 (New York: Academic Press), pp. 135–211. [4]

Hockney, R.W., and Eastwood, J.W. 1981, *Computer Simulation Using Particles* (New York: McGraw-Hill), Chapter 6. [5]

Temperton, C. 1980, *Journal of Computational Physics*, vol. 34, pp. 314–329. [6]

19.5 Relaxation Methods for Boundary Value Problems

As we mentioned in §19.0, relaxation methods involve splitting the sparse matrix that arises from finite differencing and then iterating until a solution is found.

There is another way of thinking about relaxation methods that is somewhat more physical. Suppose we wish to solve the elliptic equation

$$\mathcal{L}u = \rho \tag{19.5.1}$$