Jacobi Method

We now develop a numerical method for solving Laplace's equation. To begin, let's go back for a moment to the diffusion equation (Section 6.2). Consider the Fourier equation for the two-dimensional diffusion of temperature,

$$\frac{\partial T(x,y,t)}{\partial t} = \kappa\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) \tag{8.16}$$

where κ is the thermal diffusion coefficient. We know from physical intuition that given any initial temperature profile plus stationary boundary conditions, the solution will relax to some steady state, call it $T_s(x,y)$. In other words,

$$\lim_{t\to\infty} T(x,y,t) = T_s(x,y) \tag{8.17}$$

When the temperature profile is at the steady state, then it does not change in time, that is, $\partial T/\partial t = 0$. This means that the steady state obeys the equation

$$\frac{\partial^2 T_s}{\partial x^2} + \frac{\partial^2 T_s}{\partial y^2} = 0 \tag{8.18}$$

Does this look familiar? Of course, this is just Laplace's equation.

The idea is that the solution of Laplace's equation is just the solution of the diffusion equation in the limit $t \to \infty$. Algorithms based on this physical principal are called *relaxation methods*. We already know how to solve the diffusion equation using the FTCS scheme. We start from the two-dimensional diffusion equation

$$\frac{\partial \Phi(x,y,t)}{\partial t} = \mu\left(\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2}\right) \tag{8.19}$$

The value of the constant μ is unimportant because it drops out later. Using the FTCS scheme in two dimensions,

$$\begin{aligned}\Phi^{n+1}_{i,j} = \Phi^n_{i,j} &+ \frac{\mu\tau}{h_x^2}\{\Phi^n_{i+1,j} + \Phi^n_{i-1,j} - 2\Phi^n_{i,j}\} \\ &+ \frac{\mu\tau}{h_y^2}\{\Phi^n_{i,j+1} + \Phi^n_{i,j-1} - 2\Phi^n_{i,j}\}\end{aligned} \tag{8.20}$$

where $\Phi^n_{i,j} \equiv \Phi(x_i, y_j, t_n)$, $x_i \equiv (i-1)h_x$, $y_j \equiv (j-1)h_y$, and $t_n \equiv (n-1)\tau$. Remember that we are solving an electrostatics problem, so the potential doesn't actually depend on time. We introduce an artificial time dependence only to assist in the construction of the algorithm. A better way to interpret $\Phi^n_{i,j}$ is to call it the nth guess for the potential with (8.20) serving as a formula for improving this guess.

In Section 6.2 we saw that the FTCS scheme can be numerically unstable. In one-dimensional systems the scheme is stable if $\mu\tau/h^2 \le \frac{1}{2}$; for two-dimensional systems the scheme is stable if

$\mu\tau \quad \mu\tau \quad 1$

(see Exercise 9.5). To simplify the analysis, we take $h_x = h_y = h$ so the condition for stability is $\mu\tau/h^2 \le \frac{1}{4}$.

Since we are only interested in the steady state solution ($n \to \infty$), we want to use the largest possible time step. Setting $\mu\tau/h^2 = 1/4$, Equation (8.20) becomes

$$\Phi^{n+1}_{i,j} = \frac{1}{4}\{\Phi^n_{i+1,j} + \Phi^n_{i-1,j} + \Phi^n_{i,j+1} + \Phi^n_{i,j-1}\} \tag{8.22}$$

Notice that the diffusion constant, μ, has dropped out and that the $\Phi^n_{i,j}$ terms cancel out on the right-hand side. This equation has a good parentage; our first example of a relaxation scheme is called the *Jacobi method*. It is easy to see that the method involves replacing the value of the potential at a point with the average value of the four nearest neighbors. This result may be thought of as a discrete version of the mean-value theorem for electrostatic potential. Of course, Equation (8.22) is used only for interior points and not when (i,j) is a boundary point.

Gauss-Seidel and Simultaneous Overrelaxation

A simple modification of the Jacobi method improves its rate of convergence. Suppose that we use the updated values of $\Phi_{i,j}$ as they become available. The iteration equation is then

$$\Phi^{n+1}_{i,j} = \frac{1}{4}\{\Phi^n_{i+1,j} + \Phi^{n+1}_{i-1,j} + \Phi^n_{i,j+1} + \Phi^{n+1}_{i,j-1}\} \tag{8.23}$$

The idea is that the updated values of Φ at two of the nearest neighbors have already been computed, so why not use them. With this modification the method is called the *Gauss-Seidel* method. Besides accelerating the convergence, with Gauss-Seidel we do not need to simultaneously store both the Φ^n and Φ^{n+1} matrices, a significant savings in memory.

We can significantly improve our algorithm by overcorrecting the value of Φ at each iteration of the Gauss-Seidel method. This is achieved by using the iteration equation

$$\Phi^{n+1}_{i,j} = (1-\omega)\Phi^n_{i,j} + \frac{\omega}{4}\{\Phi^n_{i+1,j} + \Phi^{n+1}_{i-1,j} + \Phi^n_{i,j+1} + \Phi^{n+1}_{i,j-1}\} \tag{8.24}$$

where the constant ω is called the overrelaxation parameter. This method is called *simultaneous overrelaxation* (SOR).

The trick to using SOR effectively is to select a good value for ω. Notice that using $\omega = 1$ is equivalent to Gauss-Seidel. For $\omega < 1$ we have underrelaxation, and the convergence is slowed. For $\omega > 2$ the SOR method is unstable. There is an ideal value for ω between 1 and 2 that gives the best acceleration. In some geometries this optimal value is known. For example, in an $N_x \times N_y$ rectangular grid,

$2 \quad (8.25)$

Table 8.1: Outline of program `relax`, which solves Laplace's equation using the Jacobi, Gauss-Seidel, or SOR method.

- Initialize parameters (L, h, etc.).
- Select ω, the over-relaxation factor (SOR only).
- Set initial guess as first term in separation of variables solution, (8.15).
- Loop until desired fractional change per iteration is obtained.
  - Compute new estimate for Φ using:
    - * Jacobi method (8.22) **or**;
    - * Gauss-Seidel method (8.23) **or**;
    - * Simultaneous over-relaxation (SOR) method (8.24).
  - Check if fractional change is small enough to halt the iteration.
- Plot final estimate of $\Phi(x, y)$ as contour and surface plots.
- Plot the fractional change versus iteration.

See pages 268 and 271 for program listings.

where

$$r = \frac{1}{2}\left(\cos\frac{\pi}{N_x} + \cos\frac{\pi}{N_y}\right) \tag{8.26}$$

If $N_x = N_y = N$ (square geometry), this simplifies to

$$\omega_{\text{opt}} = \frac{2}{1 + \sin(\pi/N)} \tag{8.27}$$

with $\omega_{\text{opt}} \approx 1.939$ for $N = 100$. In real-life problems the optimal value for ω is obtained by empirical trial and error. Sophisticated programs will automatically adjust ω according to how well the solution is converging.

A program, called `relax`, that solves Laplace's equation using the Jacobi, Gauss-Seidel, or SOR method, is outlined in Table 8.1. Relaxation algorithms require an initial guess to start the iteration process and the `relax` program uses the first term in the separation of variables solution (8.15). Sometimes the efficiency of the algorithm is greatly influenced by the accuracy of this initial guess. The best way to appreciate this point is to run the program using a poor initial guess (e.g., $\Phi = 0$ in the interior).

The potential, as computed by the `relax` program, is illustrated in Figure 8.2 by a contour map and a mesh figure. Notice that we have no Gibbs' phenomenon (compare with the separation of variables solution, Figure 8.1).

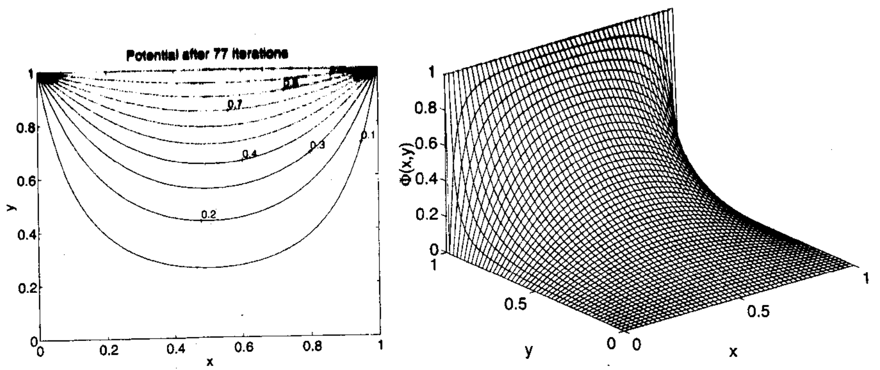

Figure 8.2: Contour and mesh plots of the potential from `relax` using the SOR method. Grid size is $N = 50$ and $\omega = 1.8$ ($\omega_{\text{opt}} \approx 1.88$). Compare with the separation of variables solution, Figure 8.1.

Poisson Equation

The methods developed so far are easy to generalize to solve the Poisson equation. In MKS units,

$$\frac{\partial^2\Phi(x,y)}{\partial x^2} + \frac{\partial^2\Phi(x,y)}{\partial y^2} = -\frac{1}{\epsilon_0}\rho(x,y) \tag{8.28}$$

where $\rho(x, y)$ is the charge density and ϵ_0 is the permittivity of free space. In discretized form, we have

$$\frac{1}{h_x^2}\{\Phi_{i+1,j} + \Phi_{i-1,j} - 2\Phi_{i,j}\} + \frac{1}{h_y^2}\{\Phi_{i,j+1} + \Phi_{i,j-1} - 2\Phi_{i,j}\} = -\frac{1}{\epsilon_0}\rho_{i,j} \tag{8.29}$$

Using the analysis presented earlier, we construct the Jacobi relaxation scheme for the Poisson equation as

$$\Phi_{i,j}^{n+1} = \frac{1}{4}\left\{\Phi_{i+1,j}^n + \Phi_{i-1,j}^n + \Phi_{i,j+1}^n + \Phi_{i,j-1}^n + \frac{1}{\epsilon_0}h^2\rho_{i,j}\right\} \tag{8.30}$$

where to simplify the formulation we take $h_x = h_y = h$. The other two schemes considered in this section may also be generalized by the simple addition of the charge density term.

EXERCISES

1. Write a program to evaluate the potential $\Phi(x, y)$ numerically, as given by Equation (8.15), on a 50×50 grid. Take $\Phi_0 = 1$ and graph your solution by mesh and contour plots (see Figure 8.1). Plot your results using terms through $n = 11$, 21, and 51. Estimate how many terms in the infinite sum are needed to obtain about 1% accuracy in the solution. [Computer]

2. (a) Find the solution to the more general boundary value problem,

$$\begin{aligned}\Phi(x=0,y) &= \Phi_1 & \Phi(x=L_x,y) &= \Phi_2\\ \Phi(x,y=0) &= \Phi_3 & \Phi(x,y=L_y) &= \Phi_4\end{aligned}$$

where $\Phi_1,\ldots,\Phi_4$ are constants. [Pencil] (b) Write a program to graph your solution by mesh and contour plots. Graph the potential for $\Phi_1=\Phi_3=1$, $\Phi_2=\Phi_4=0$, and for $\Phi_1=\Phi_2=1$, $\Phi_3=\Phi_4=0$. [Computer]

3. (a) Find the solution to the three-dimensional cubic boundary value problem, [75, Problem 2.13]

$$\begin{aligned}\Phi(x=0,y,z) &= \Phi(x=L,y,z) &= 0\\ \Phi(x,y=0,z) &= \Phi(x,y=L,z) &= 0\\ \Phi(x,y,z=0) &= \Phi(x,y,z=L) &= \Phi_0\end{aligned}$$

using separation of variables. [Pencil] (b) Write a program to graph your solution for a given height z. Produce mesh and contour plots of $\Phi(x,y,z)$ for $z=L/4$ and $L/2$. [Computer] (c) Write a three-dimensional version of the `relax` program to solve this problem by relaxation. Produce mesh and contour plots of $\Phi(x,y,z)$ for $z=L/4$ and $L/2$; compare with your results from part (b).

4. A major issue with relaxation methods is their computational speed. (a) Run the `relax` program using the Jacobi method for different-sized systems ($N_x=N_y=10$ to 50). Graph the number of iterations performed versus system size. Fit the data to a power law and approximate the exponent. (b) Repeat part (a) using a bad initial guess. Set the potential initially to zero everywhere in the interior. (c) Using SOR, repeat parts (a) and (b). Compare the Jacobi and SOR methods (use the optimum value for ω). [Computer]

5. Formulate the Jacobi method without assuming that $h_x=h_y$. Modify the `relax` program to implement this modification. Keep $L_x=L_y$ and the boundary conditions, Equation (8.7), and try grids of 32×32, 64×16, and 16×64. Do you find any significant differences? [Computer]

6. The `relax` program uses a good initial guess for the potential $\Phi(x,y)$. To illustrate its importance, run the program with a variety of initial guesses, including some poor ones (e.g., $\Phi=0$ in the interior). Also try an initial guess that uses the first few terms of the separation of variables solution (8.15). Compare and comment on your results. [Computer]

7. Modify the `relax` program to plot the electric field, $\mathbf{E}=-\nabla\Phi$. In MATLAB, the functions `gradient` (which computes the gradient) and `quiver` (which produces a field plot; see Figure 8.8) are available. Plot the electric field for the potential shown in Figure 8.2. Try both proportional and equal-length field arrows. [MATLAB]

8. Write a program that uses the SOR method to simulate a Faraday cage (Figure 8.3). Use a square geometry with $N_x=N_y=60$. Set the left and right walls to $\Phi=0$ and $\Phi=100$, respectively. Fix the potential at the top and bottom walls but have it vary linearly across the system. (a) The Faraday cage is represented by the following eight points: $(i,j)=(20,20)$, $(30,20)$, $(40,20)$, $(20,30)$, $(20,40)$, $(30,40)$, $(40,30)$, and $(40,40)$. The potential at these points is fixed at zero. Plot the potential $\Phi_{i,30}$ versus i (i.e., a horizontal cross section through the center), both with and without the cage. (b) Try a cage that has only the four corner points $(20,20)$, $(20,40)$, $(40,20)$, $(40,40)$, and compare with the results from part (a). (c) Try a cage that has only the four side points $(20,30)$, $(30,20)$, $(40,30)$, $(30,40)$, and compare with the results from part (a). [Computer]

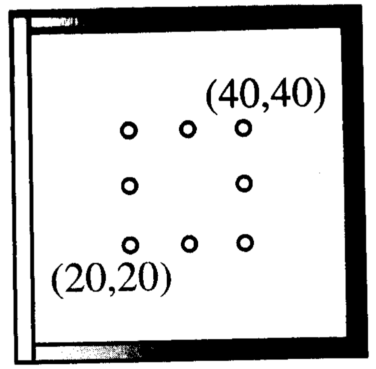

Figure 8.3: Faraday cage.

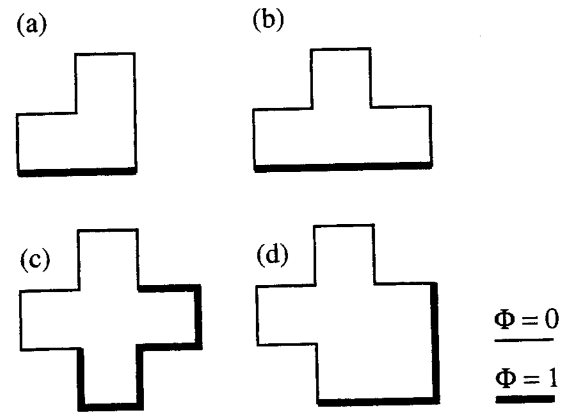

Figure 8.4: Geometries and boundary conditions for Laplace equation.

9. Write a program that uses the SOR method to solve the electrostatics problems shown in Figure 8.4. For each box, the thin lines indicate a boundary where the potential is fixed at zero; a thick line indicates the potential is fixed at one. [Computer]

10. (a) Write a program that solves the two-dimensional Poisson equation in a square geometry with the Dirichlet boundary conditions $\Phi=0$ at the boundary. Map the potential for a single charge at the center of the system. Compare with the potential for a charge in free space. Remember that in two dimensions this charge is a line charge and not a point charge (see Figure 8.5). (b) Modify your program to use periodic boundary conditions. Compare with the results from part (a). (Hint: Think.) [Computer]