Chapter 9

The Curse of Dimensionality

9.1 Two and Three Space Dimensions

In the previous chapter we considered partial differential equations in two independent variables: time and one space variable. Since physical phenomena occur in a three-dimensional world, mathematical models in only one space dimension are usually considerable simplifications of the actual physical situation although in many cases they are sufficient for phenomena that exhibit various symmetries or in which events are happening in two of the three space dimensions at such a slow rate that those directions can be ignored. However, large-scale scientific computing is now increasingly concerned with more detailed analyses of problems in which all three space directions, or at least two, are of concern. This chapter, then, will be concerned with problems in more than one space dimension, although for simplicity of exposition we will mainly discuss only two-dimensional problems.

In the previous chapter we considered the heat equation

$$u_t = cu_{xx} \tag{9.1.1}$$

as a mathematical model of the temperature in a long, thin rod. If the body of interest is a three-dimensional cube, as shown in Figure 9.1, (9.1.1) extends to three dimensions with partial derivatives in all three variables x, y, and z. Thus

$$u_t = c(u_{xx} + u_{yy} + u_{zz}), \tag{9.1.2}$$

where the constant c is again the ratio of the thermal conductivity and the product of the specific heat and mass density under the assumption that these quantities are not functions of space or time.

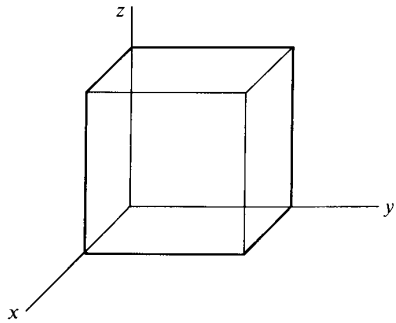

Figure 9.1: *Three-Dimensional Cube*

Equation (9.1.2) is a model of the temperature u as a function of time and at points within the interior of the body. As usual, to complete the model we need to specify boundary conditions, and for this purpose it is simplest for exposition to treat the corresponding problem in two space dimensions:

$$u_t = c(u_{xx} + u_{yy}). \tag{9.1.3}$$

We can consider (9.1.3) to be the mathematical model of the temperature in a flat, thin plate as shown in Figure 9.2, where we have taken the plate to be the unit square.

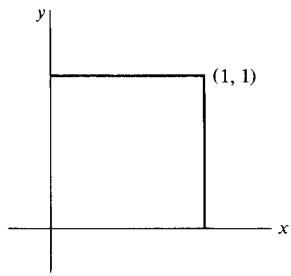

Figure 9.2: *Flat, Thin Plate*

The simplest boundary conditions occur when the temperature is prescribed on the four sides of the plate:

$$u(t, x, y) = g(x, y), \qquad (x, y) \text{ on boundary}, \tag{9.1.4}$$

where g is a given function. Another possibility is to assume that one of the sides, say $x = 0$, is perfectly insulated; thus, there is no heat loss across that side and no change in temperature, so the boundary condition is

$$u_x(t, 0, y) = 0, \qquad 0 \leq y \leq 1, \tag{9.1.5}$$

combined with the specification (9.1.4) on the other sides. A boundary condition of the form (9.1.5) is usually called a *Neumann condition*, and that of the form (9.1.4) is a *Dirichlet condition.* Clearly, various other such combinations are possible, including a specified temperature change (other than zero) across a boundary. Boundary conditions for the three-dimensional problem can be given in a similar fashion. We also must specify a temperature distribution at some time which we take to be $t = 0$; such an initial condition for (9.1.3) is of the form

$$u(0, x, y) = f(x, y). \tag{9.1.6}$$

Given the initial condition (9.1.6) and boundary conditions of the form (9.1.4) and/or (9.1.5), it is intuitively clear that the temperature distribution should evolve in time to a final steady state that is determined only by the boundary conditions. In many situations it is this steady-state solution that is of primary interest, and since it no longer depends on time it should satisfy the equation (9.1.3) with $u_t = 0$:

$$u_{xx} + u_{yy} = 0. \tag{9.1.7}$$

This is Laplace's equation and, as mentioned in the previous chapter, is the prototype of an elliptic equation. If we wish only the steady-state solution of the temperature distribution problem that we have been discussing, we can proceed, in principle, in two ways: solve equation (9.1.3) for u as a function of time until convergence to a steady state is reached, or solve (9.1.7) only for the steady-state solution.

Finite Differences for Poisson's Equation

We will return to the time-dependent problem shortly, after considering the finite difference method for (9.1.7) and, more generally, Poisson's equation

$$u_{xx} + u_{yy} = f, \tag{9.1.8}$$

where f is a given function of x and y. We assume that the domain of the problem is the unit square $0 \leq x, y \leq 1$, and that Dirichlet boundary conditions

$$u(x, y) = g(x, y), \qquad (x, y) \text{ on boundary} \tag{9.1.9}$$

are given, where g is a known function. We impose a mesh of grid points on the unit square with spacing h between the points in both the horizontal and vertical directions; this is illustrated in Figure 9.3.

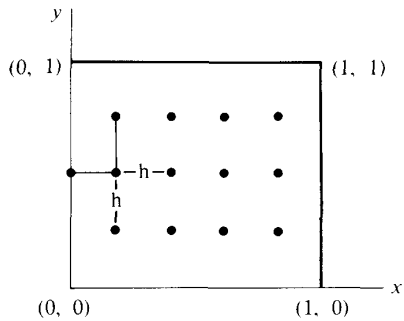

Figure 9.3: *Mesh Points on the Unit Square and Discretization Stencil*

The interior grid points are given by

$$(x_i, y_j) = (ih, jh), \qquad i, j = 1, \ldots N, \tag{9.1.10}$$

where $(N+1)h = 1$. Now consider a typical grid point (x_i, y_j). We approximate u_{xx} and u_{yy} at this point by the centered difference approximations

$$u_{xx}(x_i, y_j) \doteq \frac{1}{h^2}[u(x_{i-1}, y_j) - 2u(x_i, y_j) + u(x_{i+1}, y_j)], \tag{9.1.11a}$$

$$u_{yy}(x_i, y_j) \doteq \frac{1}{h^2}[u(x_i, y_{j-1}) - 2u(x_i, y_j) + u(x_i, y_{j+1})]. \tag{9.1.11b}$$

If we put these approximations into the differential equation (9.1.8), we obtain

$$\begin{aligned} u(x_{i-1}, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 4u(x_i, y_j) \\ \doteq h^2 f(x_i, y_j), \end{aligned} \tag{9.1.12}$$

which is an approximate relationship that the exact solution u of (9.1.8) satisfies at any grid point in the interior of the domain.

We now define approximations u_{ij} to the exact solution $u(x_i, y_j)$ at the N^2 interior grid points by requiring that they satisfy exactly the relationship (9.1.12); that is,

$$-u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} + 4u_{ij} = -h^2 f_{ij}, \qquad i, j = 1, \ldots, N, \tag{9.1.13}$$

where we have multiplied (9.1.12) by -1. This is a linear system of equations in the $(N+2)^2$ variables u_{ij}. Note, however, that the variables $u_{0,j}$, $u_{N+1,j}$, $j = 0, \ldots, N+1$, and $u_{i,0}$, $u_{i,N+1}$, $i = 0, \ldots, N+1$, correspond to the grid points on the boundary and thus are given by the boundary condition (9.1.9):

$$\begin{aligned} u_{0,j} &= g(0, y_j), \qquad & u_{N+1,j} &= g(1, y_j), \qquad & j &= 0, 1, \ldots, N+1, \\ u_{i,0} &= g(x_i, 0), \qquad & u_{i,N+1} &= g(x_i, 1), \qquad & i &= 0, 1, \ldots, N+1. \end{aligned} \tag{9.1.14}$$

Therefore (9.1.13) is a linear system of N^2 equations in the N^2 unknowns u_{ij}, $i,j = 1,\ldots,N$, corresponding to the interior grid points. The stencil in Figure 9.3 shows how u_{ij} is coupled to its north, south, east, and west neighbors in (9.1.13). It is easy to show (Exercise 9.1.1) that the local discretization error in the u_{ij} is $0(h^2)$. Note that (9.1.13) is the natural extension to two space variables of the discrete equations

$$-u_{i+1} + 2u_i - u_{i-1} = -h^2 f_i, \qquad i = 1,\ldots,N,$$

obtained in Chapter 3 for the "one-dimensional Poisson equation" $u'' = f$.

We now wish to write the system (9.1.13) in matrix-vector form, and for this purpose we will number the interior grid points in the manner shown in Figure 9.4, which is called the *natural* or *row-wise ordering*. Corresponding to this ordering of the grid points, we order the unknowns $\{u_{ij}\}$ into the vector

$$(u_{11},\ldots,u_{N1},u_{12},\ldots,u_{N2},\ldots,u_{1N},\ldots,u_{NN}), \tag{9.1.15}$$

and write the system of equations in the same order. We illustrate this for $N = 2$ (Exercise 9.1.2):

$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_{11} \\ u_{21} \\ u_{12} \\ u_{22} \end{bmatrix} = -h^2 \begin{bmatrix} f_{11} \\ f_{21} \\ f_{12} \\ f_{22} \end{bmatrix} + \begin{bmatrix} u_{01} + u_{10} \\ u_{20} + u_{31} \\ u_{02} + u_{13} \\ u_{32} + u_{23} \end{bmatrix}, \tag{9.1.16}$$

in which we have put the known boundary values on the right-hand side of the equation.

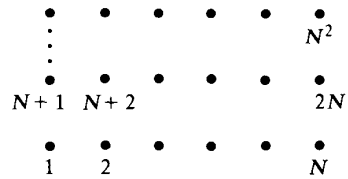

Figure 9.4: *Natural Ordering of the Interior Grid Points*

The equations (9.1.16) begin to illustrate the structure of the linear system. For general N, a typical row of the matrix will be

$$-1 \quad 0 \quad \cdots \quad 0 \quad -1 \quad 4 \quad -1 \quad 0 \quad \cdots \quad 0 \quad -1$$

where $N-2$ zeros separate the -1's in both directions. The equations corresponding to an interior grid point adjacent to a boundary point will contain a known boundary value, and this value will be moved to the right side of the

$$
\begin{array}{cccccccccccccccc}
4 & -1 & & & -1 & & & & & & & & & & & \\
-1 & 4 & -1 & & & -1 & & & & & & & & & & \\
 & -1 & 4 & -1 & & & -1 & & & & & & & & & \\
 & & -1 & 4 & & & & -1 & & & & & & & & \\
-1 & & & & 4 & -1 & & & -1 & & & & & & & \\
 & -1 & & & -1 & 4 & -1 & & & -1 & & & & & & \\
 & & -1 & & & -1 & 4 & -1 & & & -1 & & & & & \\
 & & & -1 & & & -1 & 4 & & & & -1 & & & & \\
 & & & & -1 & & & & 4 & -1 & & & -1 & & & \\
 & & & & & -1 & & & -1 & 4 & -1 & & & -1 & & \\
 & & & & & & -1 & & & -1 & 4 & & & & -1 & \\
 & & & & & & & -1 & & & & 4 & -1 & & & \\
 & & & & & & & & -1 & & & -1 & 4 & -1 & & \\
 & & & & & & & & & -1 & & & -1 & 4 & -1 & \\
 & & & & & & & & & & -1 & & & -1 & 4 &
\end{array}
$$

Figure 9.5: *Coefficient Matrix of* (9.1.13) *for* $N = 4$

equation, eliminating the corresponding -1 from the matrix. This happened in each equation in (9.1.16) because of the size of N. We show in Figure 9.5 the coefficient matrix for $N = 4$ (see Exercise 9.1.2).

Although Figure 9.5 illustrates the structure of the coefficient matrix, this form is cumbersome for large N; it is much easier to write it in a block matrix form. To do this we define the $N \times N$ tridiagonal matrix

$$
T_N = \begin{bmatrix}
4 & -1 & & & \\
-1 & \ddots & \ddots & & \\
 & \ddots & \ddots & & \\
 & & \ddots & & -1 \\
 & & & -1 & 4
\end{bmatrix} \tag{9.1.17}
$$

and let I_N denote the $N \times N$ identity matrix. Then the $N^2 \times N^2$ coefficient matrix of (9.1.13) is the *block tridiagonal matrix*

$$
A = \begin{bmatrix}
T_N & -I_N & & & \\
-I_N & T_N & \ddots & & \\
 & -I_N & \ddots & & \\
 & & \ddots & & -I_N \\
 & & & -I_N & T_N
\end{bmatrix}. \tag{9.1.18}
$$

The matrix of (9.1.16) is the special case of (9.1.18) for $N = 2$, and Figure 9.5 shows the matrix for $N = 4$.

If we also define the vectors

$$\mathbf{u}_i = (u_{1i}, \ldots, u_{Ni})^T, \qquad \mathbf{f}_i = (f_{1i}, \ldots, f_{Ni})^T, \qquad i = 1, \ldots, N,$$
$$\mathbf{b}_1 = (u_{01} + u_{10}, u_{20}, \ldots, u_{N-1,0}, u_{N,0} + u_{N+1,1})^T,$$
$$\mathbf{b}_i = (u_{0i}, 0, \ldots, 0, u_{N+1,i})^T, \qquad i = 2, \ldots, N-1,$$
$$\mathbf{b}_N = (u_{0,N} + u_{1,N+1}, u_{2,N+1}, \ldots, u_{N-1,N}, u_{N,N+1} + u_{N+1,N})^T,$$

then we can write the system (9.1.13) in the compact form

$$\begin{bmatrix} T_N & -I_N & & \\ -I_N & \ddots & \ddots & \\ & \ddots & & -I_N \\ & & -I_N & T_N \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_N \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 - h^2\mathbf{f}_1 \\ \mathbf{b}_2 - h^2\mathbf{f}_2 \\ \vdots \\ \mathbf{b}_N - h^2\mathbf{f}_N \end{bmatrix}. \tag{9.1.19}$$

We now make several comments about this system of equations. If N is of moderate size, say $N = 100$, then there are $N^2 = 10^4$ unknowns, and the matrix in (9.1.19) is $10,000 \times 10,000$. In each row of the matrix there are at most five nonzero elements, regardless of the size of N, so the distribution of nonzero to zero elements is very "sparse" if N is at all large. Such matrices are called *large sparse matrices* and arise in a variety of ways besides the numerical solution of partial differential equations.

It is the property of being sparse that allows such large systems of equations to be solved on today's computers with relative ease. Recall that in Chapter 4, we saw that Gaussian elimination requires on the order of n^3 arithmetic operations to solve an $n \times n$ linear system. Hence if a $10^4 \times 10^4$ linear system were "dense," that is, few of its elements were zero, and Gaussian elimination were used to solve the system, then on the order of 10^{12} operations would be required. At a rate of 10^6 operations per second, it would require several hours to solve such a system. Moreover, for the corresponding three dimensional problem the size of the system would be 10^6, requiring 10^{18} operations, which is completely beyond the capacity of the fastest computers. However, by utilizing the special structure and sparsity of systems such as (9.1.19), we shall see in the next two sections that they can be accurately solved relatively quickly and accurately, despite their large size.

The Heat Equation

We end this section by applying the discretization of Poisson's equation to the heat equation (9.1.3) in two space variables where, again for simplicity in exposition, we will assume that the x, y domain is the unit square of Figure 9.3

and that the Dirichlet boundary conditions (9.1.4) are given on the sides of the square. We also assume the initial condition (9.1.6).

Corresponding to the method (8.2.5) in the case of a single space variable, we can consider the following explicit method for (9.1.3):

$$u_{ij}^{m+1} = u_{ij}^m + \frac{c\Delta t}{h^2}(u_{i,j+1}^m + u_{i,j-1}^m + u_{i+1,j}^m + u_{i-1,j}^m - 4u_{ij}^m), \tag{9.1.20}$$

for $m = 0, 1, \ldots$, and $i, j = 1, \ldots, N$. Here u_{ij}^m denotes the approximate solution at the i, j gridpoint and at the mth time level $m\Delta t$, and u_{ij}^{m+1} is the approximate solution at the next time level. The terms in parentheses on the right-hand side of (9.1.20) correspond exactly to the discretization (9.1.13) with $f_{ij} = 0$. The prescription (9.1.20) has the same properties as its one-dimensional counterpart (8.2.5): it is first-order accurate in time and second-order accurate in space, and it is easy to carry out. It is also subject to a similar stability condition

$$\Delta t \leq \frac{h^2}{4c}, \tag{9.1.21}$$

and thus has the problem that if h is small, very small time steps are required.

We can attempt to circumvent this restriction on the time step in the same way that we did in Section 8.3 by the use of implicit methods. For example, the implicit method (8.3.2) now becomes

$$u_{ij}^{m+1} = u_{ij}^m + \frac{c\Delta t}{h^2}(u_{ij+1}^{m+1} + u_{ij-1}^{m+1} + u_{i+1,j}^{m+1} + u_{i-1,j}^{m+1} - 4u_{ij}^{m+1}), \tag{9.1.22}$$

which is unconditionally stable. However, to carry out this method requires the solution at each time step of the system of linear equations

$$\left(4 + \frac{h^2}{c\Delta t}\right) u_{ij}^{m+1} - u_{ij+1}^{m+1} - u_{ij-1}^{m+1} - u_{i+1,j}^{m+1} - u_{i-1,j}^{m+1} = \frac{h^2}{c\Delta t} u_{ij}^m, \tag{9.1.23}$$

for $i, j = 1, \ldots, N$. This system has the same form as the system (9.1.13) for Poissson's equation, with the exception that the coefficent of u_{ij}^{m+1} is modified. Indeed, the left-hand sides of the equations (9.1.23) have exactly the same form as the finite difference equations for the *Helmholtz equation* $u_{xx} + u_{yy} - \sigma u = 0$, where σ is a given function of x and y. The term $-\sigma u$ in this differential equation becomes $-h^2\sigma_{ij}u_{ij}$ in the difference equations (9.1.13), with $\sigma_{ij} = \sigma(x_i, y_j)$. The particular constant function $\sigma = -1/(c\Delta t)$ then corresponds to the left-hand side of (9.1.23).

In the case of a single space variable, the use of an implicit method such as (8.3.2) does not cause much computational difficulty since the solution of tridiagonal systems of equations can be accomplished so rapidly. However, each time step of (9.1.23) requires the solution of a two-dimensional Poisson-type equation, which is a much more difficult computational problem. The

Crank-Nicolson method (8.3.12) can also be easily extended to equation (9.1.3) (Exercise 9.1.3) but suffers from the same difficulty that Poisson-type equations must be solved at each time step. We shall consider, instead, a different class of methods, in which the basic computational step is the solution of tridiagonal systems of equations. These are *time-splitting* methods, in which the time interval $(t, t+\Delta t)$ is further subdivided and, in essence, only one-dimensional problems are solved at each timestep.

An Alternating Direction Method

One of the most classical time-splitting methods is the *Peaceman-Rachford alternating direction implicit (ADI)* method, which has the form

$$u_{ij}^{m+1/2} = u_{ij}^{m} + \frac{1}{2}\frac{c\Delta t}{h^2}(u_{i+1,j}^{m+1/2} + u_{i-1,j}^{m+1/2} - 2u_{ij}^{m+1/2} + u_{i,j+1}^{m} + u_{i,j-1}^{m} - 2u_{ij}^{m}), \tag{9.1.24a}$$

$$u_{ij}^{m+1} = u_{ij}^{m+1/2} + \frac{1}{2}\frac{c\Delta t}{h^2}(u_{i,j+1}^{m+1} + u_{i,j-1}^{m+1} - 2u_{ij}^{m+1} + u_{i+1,j}^{m+1/2} + u_{i-1,j}^{m+1/2} - 2u_{ij}^{m+1/2}). \tag{9.1.24b}$$

This is a two-step method in which intermediate values $u_{ij}^{m+1/2}$, $i, j = 1, \ldots, N$, are computed at the first step (9.1.24a). These $u_{ij}^{m+1/2}$ are to be interpreted as approximate values of the solution at the intermediate time level $m + \frac{1}{2}$; thus the factor $\frac{1}{2}$ appears on the right-hand side of (9.1.24a) because the time step is $\frac{1}{2}\Delta t$. The computation in (9.1.24a) involves the solution of the N tridiagonal systems of equations

$$\begin{aligned}(2+\alpha)u_{ij}^{m+1/2} - u_{i+1,j}^{m+1/2} - u_{i-1,j}^{m+1/2} \\ = (\alpha - 2)u_{ij}^{m} + u_{i,j+1}^{m} + u_{i,j-1}^{m}, \qquad i = 1, \ldots, N,\end{aligned} \tag{9.1.25}$$

for $j = 1, \ldots, N$, where $\alpha = 2h^2/(c\Delta t)$; that is, for each fixed j, (9.1.25) is a tridiagonal system whose solution is $u_{ij}^{m+1/2}$, $i = 1, \ldots, N$. The coefficient matrix of each of these systems is $\alpha I + A$, where A is the $(2, -1)$ tridiagonal matrix (3.1.10). Since $\alpha I + A$ is diagonally dominant, these systems can be rapidly solved by Gaussian elimination without interchanges.

Once the intermediate values $u_{ij}^{m+1/2}$ have been computed, the final values u_{ij}^{m+1} are obtained from (9.1.24b) by solving the N tridiagonal systems

$$\begin{aligned}(2+\alpha)u_{ij}^{m+1} - u_{i,j+1}^{m+1} - u_{i,j-1}^{m+1} \\ = (\alpha - 2)u_{ij}^{m+1/2} + u_{i+1,j}^{m+1/2} + u_{i-1,j}^{m+1/2}, \qquad j = 1, \ldots, N,\end{aligned} \tag{9.1.26}$$

for $i = 1, \ldots, N$. Again, the coefficient matrices of these systems are $\alpha I + A$. Thus the computational process requires the solution of $2N$ tridiagonal systems of dimension N to move from the mth time level to the $(m + 1)$st. It can be shown that this method is unconditionally stable.

The term *alternating direction* derives from the paradigm that in some sense we are approximating values of the solution in the x-direction by (9.1.24a), and then in the y-direction by (9.1.24b). There are many variants of ADI methods and, more generally, of time-splitting methods, and such methods are widely used for parabolic-type equations.

Supplementary Discussion and References: 9.1

The discussion of this section has been restricted to Poisson's equation in two variables on a square domain and the corresponding heat equation. However, problems that arise in practice will generally deviate considerably from these ideal conditions: the domain may not be square; the equation may have nonconstant coefficients or even be nonlinear; the boundary conditions may be a mixture of Dirichlet and Neumann conditions; there may be more than a single equation – that is, there may be a coupled system of partial differential equations; the equations may have derivatives of order higher than two; and there may be three or more independent variables. The general principles of finite difference discretization of this section still apply, but each of the preceding factors causes complications.

One of the classical references for the discretization of elliptic equations by finite difference methods is Forsythe and Wasow [1960]. See also Roache [1972] for problems that arise in fluid dynamics, and Hall and Porsching [1990]. Discussions and analyses of alternating direction methods and related methods such as the method of fractional steps are found in a number of books; see, for example, Varga [1962] and Richtmyer and Morton [1967].

In the last several years finite element and other projection-type methods have played an increasingly important role in the solution of elliptic- and parabolic-type equations. Although the mathematical basis of the finite element method goes back to the 1940s, its development into a viable procedure was carried out primarily by engineers in the 1950s and 1960s, especially for problems in structural analysis. Since then the mathematical basis has been extended and broadened and its applicability to general elliptic and parabolic equations well demonstrated. One of the method's main advantages is its ability to handle curved boundaries. For introductions to the finite element method, see Strang and Fix [1973], Becker, Carey and Oden [1981], Carey and Oden [1984], Axelsson and Barker [1984], and Hall and Porsching [1990].

EXERCISES 9.1

9.1.1. Assume that the function u is as many times continuously differentiable as needed. Expand u in Taylor series about (x_i, y_j) and show that the approximations (9.1.11) are second order accurate:

$$u_{xx}(x_i, y_j) - \frac{1}{h^2}[u(x_{i-1}, y_j) - 2u(x_i, y_j) + u(u_{i+1}, y_j)] = 0(h^2),$$

and similarly for the approximation to u_{yy}. Conclude that the local discretization error in the approximate solution of (9.1.13) is $0(h^2)$.

9.1.2. Verify that with the ordering (9.1.15), the system of equations (9.1.13) takes the form (9.1.16) for $N = 2$. Verify also that Figure 9.5 gives the coefficient matrix for $N = 4$. What is the right-hand side of the system in this case?

9.1.3. Formulate the Crank-Nicolson method for (9.1.3).

9.1.4. Consider the equation

$$u_{xx} + u_{yy} + \sigma u_x = f$$

with the boundary conditions (9.1.9) on the unit square and with σ a positive constant. Approximate u_x by central differences:

$$u_x(x, y) \doteq \frac{u(x + h, y) - u(x - h, y)}{2h}.$$

Write the difference equations corresponding to (9.1.13). Under what condition on σh will the coefficient matrix be diagonally dominant?

9.2 Direct Methods

In the previous section we obtained the system of linear equations (9.1.19) after discretizing Poisson's equation. We now consider ways to solve such systems. In the next section we study iterative methods; in the present section, we treat Gaussian elimination and other direct methods.

Fill in Gaussian Elimination

The coefficient matrix of (9.1.19) is banded with semi bandwidth N, but it is very sparse within the band: whatever the size of N, there are at most five non-zero elements in each row. However, if we apply Gaussian elimination or Cholesky factorization to this banded system, almost all elements that were zero within the band in the original matrix will become non-zero as the factorization proceeds. Such non-zero elements are called *filled-in elements* or simply *fill*. The way that fill occurs can be seen by examining the basic step in Gaussian elimination (see Section 4.2):

$$a_{ij}^{(k+1)} = [a_{ij}^{(k)} - a_{ik}^{(k)} a_{kj}^{(k)}]/a_{kk}^{(k)}. \tag{9.2.1}$$

At the kth stage of the elimination, the element $a_{ij}^{(k)}$ is modified to become $a_{ij}^{(k+1)}$, as shown in (9.2.1). If $a_{ij}^{(k)}$ is zero but $a_{ik}^{(k)}$ and $a_{kj}^{(k)}$ are both non-zero, $a_{ij}^{(k+1)}$ is non-zero and fill will occur in the i, j position at this stage. The more fill that occurs, the higher the operation count since elements that have become non-zero have to be eliminated later in the process. Ideally no fill would occur, so that the operation count would be based on the original number of non-zero elements in A.

In many, if not most, problems it is difficult to ascertain where fill will occur without carrying out the elimination process. In some cases, however, especially where A has a simple structure, it is possible to determine the fill pattern easily. We next do this for the matrix of (9.1.19). We will consider only the block 3×3 form of this matrix since this clearly exhibits the fill pattern. It is convenient to do the analysis in terms of the LU decomposition of A; if we partition L and U corresponding to A we have:

$$\begin{bmatrix} T & -I & \\ -I & T & -I \\ & -I & T \end{bmatrix} = \begin{bmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} & \\ & U_{22} & U_{23} \\ & & U_{33} \end{bmatrix}. \tag{9.2.2}$$

Equating the corresponding submatrices in (9.2.2), gives

$$L_{11}U_{11} = T, \tag{9.2.3a}$$

$$L_{21}U_{11} = -I \text{ or } L_{21} = -U_{11}^{-1}; \quad U_{12} = -L_{11}^{-1}, \tag{9.2.3b}$$

$$L_{22}U_{22} = T - L_{21}U_{12} = T - U_{11}^{-1}L_{11}^{-1} = T - T^{-1}, \tag{9.2.3c}$$

$$L_{32} = -U_{22}^{-1}, \qquad U_{23} = -L_{22}^{-1}, \tag{9.2.3d}$$

$$L_{33}U_{33} = T - L_{32}U_{23} = T - U_{22}^{-1}L_{22}^{-1}. \tag{9.2.3e}$$

L_{11} and U_{11} are the LU factors of the tridiagonal matrix T. By (4.2.18), these factors have the form

$$L_{11} = \begin{bmatrix} 1 & & & \\ l_2 & 1 & & \\ & \ddots & \ddots & \\ & & l_N & 1 \end{bmatrix}, \; U_{11} = \begin{bmatrix} u_1 & -1 & & \\ & u_2 & \ddots & \\ & & \ddots & -1 \\ & & & u_N \end{bmatrix}, \tag{9.2.4}$$

where the -1's in U_{11} are the off-diagonal elements of T. Even though L_{11} has only two non-zero diagonals, the same is not true of L_{11}^{-1}; it is a full lower triangular matrix. To see why this is true, recall from (4.2.20) that the ith column of L_{11}^{-1} is the solution of the system

$$L_{11}\mathbf{x}_i = \mathbf{e}_i, \tag{9.2.5}$$

where $\mathbf{e}_i$ is the vector with 1 in the ith position and zero elsewhere. The solution of (9.2.5) for $i = 1$ is

$$x_1 = 1, x_2 = -l_2 x_1 = -l_2, x_3 = -l_3 x_2 = l_2 l_3, \cdots, x_N = \pm l_2 \cdots l_N.$$

Thus provided that none of the l_i is zero, which is the case if L_{11} is the factor of T, all components of the first column of L_{11}^{-1} are non-zero. Doing the analogous computation for general i, one sees that the first non-zero component in the solution is the ith position and then each subsequent component of the solution is non-zero. It follows that each column of L_{11}^{-1} has all non-zero elements below the main diagonal. The same is true for U_{11}^T so that U_{11}^{-1} is full above the main diagonal. It is easy to verify (Exercise 9.2.1) that the product $U_{11}^{-1} L_{11}^{-1}$ is then a completely full matrix, and therefore the factors L_{22} and U_{22} in (9.2.3c) are full below and above the main diagonal, respectively. The same is true of the factors L_{32}, L_{33}, U_{23}, and U_{33}. Thus the non-zero structure of the factor L of (9.2.2) is as shown in Figure 9.6.

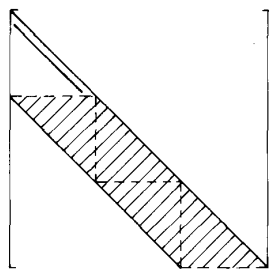

Figure 9.6: *Non-zero Structure of L and* U^T

U^T has the same structure and complete fill has occurred within the band, except for the first block. The same is true no matter how many blocks are in the matrix A; the 3×3 block structure of (9.2.2) was just used as an example. Thus, the amount of work to carry out the LU factorization (or Cholesky factorization, where the same fill pattern occurs) is almost as much as if the matrix A were a full banded matrix: the sparse structure of A within the band has essentially been lost because of the fill. This phenomenon is not dependent upon the particular $4, -1$ entries in A, and is true in much more general situations. Moreover, the QR factorization will also suffer from the same type of fill (Exercise 9.2.7).

One approach to circumventing this problem of fill is a reordering of the equations and unknowns. Consider the matrix with the non-zero structure of Figure 9.7(a). If Gaussian elimination is applied to this matrix, all elements will, in general, fill. On the other hand, for the matrix of Figure 9.7.(b) no elements will fill. The matrix of Figure 9.7(b) may be obtained from that

of 9.7(a) by a reordering of the unknowns and equations (Exercise 9.2.2). In

(a) (b)

Figure 9.7: *Arrowhead Matrices*

general, it will not be known in advance how to do a reordering that will minimize fill, but algorithms are known that can approximate this; see the Supplementary Discussion.

Domain Decomposition Reordering

We now consider a way to order the systems of equations (9.1.13) for the discrete Poisson problem so that Gaussian elimination can be carried out with less fill, and therefore fewer arithmetic operations, than if we used the natural ordering. We consider for illustration a rectangular grid of 22 interior points, as shown in Figure 9.8. We partition this grid into three subdomains, as well as two vertical lines of grid points called the *separator set*, labeled S. Such a partitioning is an example of a *domain decomposition.* We next number the grid points in the first subdomain using the natural ordering, followed by the points in the second, and third subdomains, and then finally those in the separator set. This is illustrated by the grid point numbers in the example of Figure 9.8.

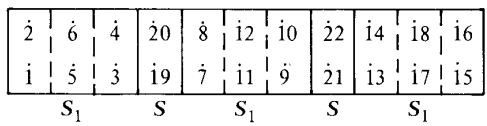

Figure 9.8: *Domain Decomposition*

We now order the equations and unknowns according to the grid point numbering of Figure 9.8. The resulting coefficient matrix is shown in Figure 9.9. Also shown in Figure 9.9 is the fill pattern that results from Gaussian elimination (or Cholesky factorization). The original elements of the matrix are 4

and -1. The other integers i indicate elements that were zero in the original matrix but have become non-zero when elements in the ith column are zeroed in the Gaussian elimination process. It is left to Exercise 9.2.3 to verify the details of Figure 9.9. We note that the matrix (before fill) of Figure 9.9 is related by a permutation matrix to the matrix that would result from using the natural ordering (Exercise 9.2.4).

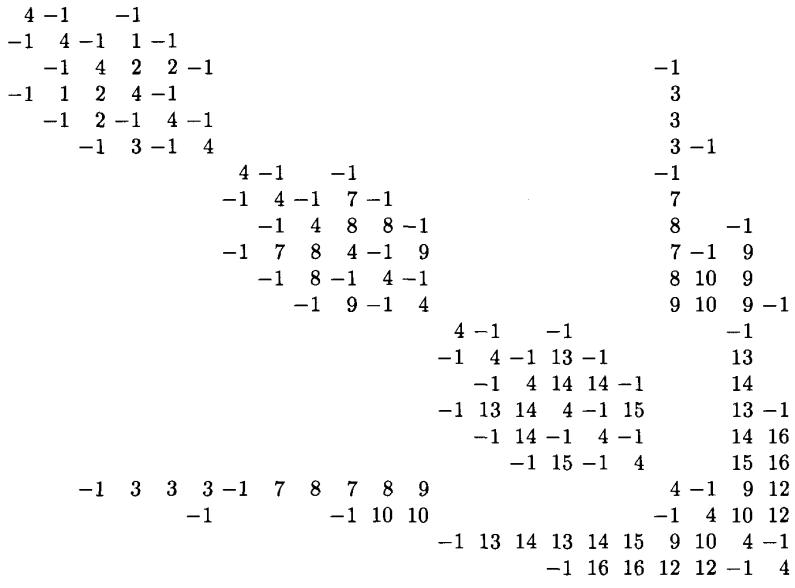

Figure 9.9: *Domain Decomposition Matrix and Fill Pattern*

The above discussion has illustrated in a very simple case the principle of domain decomposition. More generally, if we have p subdomains and separator sets that prevent unknowns in one subdomain from being connected to unknowns in any other subdomain, then the coefficient matrix will take the *block arrowhead matrix* form,

$$A = \begin{bmatrix} A_1 & & & & B_1^T \\ & A_2 & & & B_2^T \\ & & \ddots & & \vdots \\ & & & A_p & B_p^T \\ B_1 & B_2 & \cdots & B_p & A_s \end{bmatrix}. \tag{9.2.6}$$

Figure 9.9 (without the fill) is the special case $p = 3$. This domain decomposition process can be applied to more general partial differential equations, more general domains, and more general discretizations.

Now assume that the matrix of (9.2.6) has an LU factorization. Then the factors may be written in the form (Exercise 9.2.5)

$$L = \begin{bmatrix} L_1 & & & & \\ & L_2 & & & \\ \vdots & & \ddots & & \\ & & & L_p & \\ \hat{L}_1 & \hat{L}_2 & \cdots & \hat{L}_p & L_s \end{bmatrix} \quad U = \begin{bmatrix} U_1 & & \cdots & & \hat{U}_1 \\ & U_2 & & & \hat{U}_2 \\ & & \ddots & & \vdots \\ & & & U_p & \hat{U}_p \\ & & & & U_s \end{bmatrix}, \quad (9.2.7)$$

where

$$L_i U_i = A_i, \qquad i = 1, \ldots, p, \quad (9.2.8a)$$

$$\hat{L}_i U_i = B_i, \qquad L_i \hat{U}_i = B_i^T, \qquad i = 1, \ldots, p, \quad (9.2.8b)$$

$$L_s U_s = A_s - \sum_{i=1}^{p} \hat{L}_i \hat{U}_i. \quad (9.2.8c)$$

Thus the factors L and U have the same block structure as A itself. All fill occurs only within these blocks and not outside them, as illustrated in Figure 9.9.

2	6	4	20	8	12	10	22	14	18	16
1	5	3	19	7	11	9	21	13	17	15
	S_1		S		S_1		S		S_1	

Figure 9.10: *Nested Dissection Reordering of Grid Points*

Nested Dissection

We return to the example of Figure 9.9. The number of fill elements shown in the figure is 76. This is to be contrasted with 182 fill elements had we used the natural ordering and applied Gaussian elimination to the resulting banded matrix (Exercise 9.2.6). We can obtain still further improvement in the amount of fill by applying the domain decomposition principle again to each subdomain of Figure 9.8. The resulting numbering of the grid points is shown in Figure 9.10 and the corresponding coefficient matrix with fill elements in Figure 9.11. The ordering of Figure 9.8 is sometimes called *one-way dissection*, and repeated use of this within the subdomains leads to *nested dissection*.

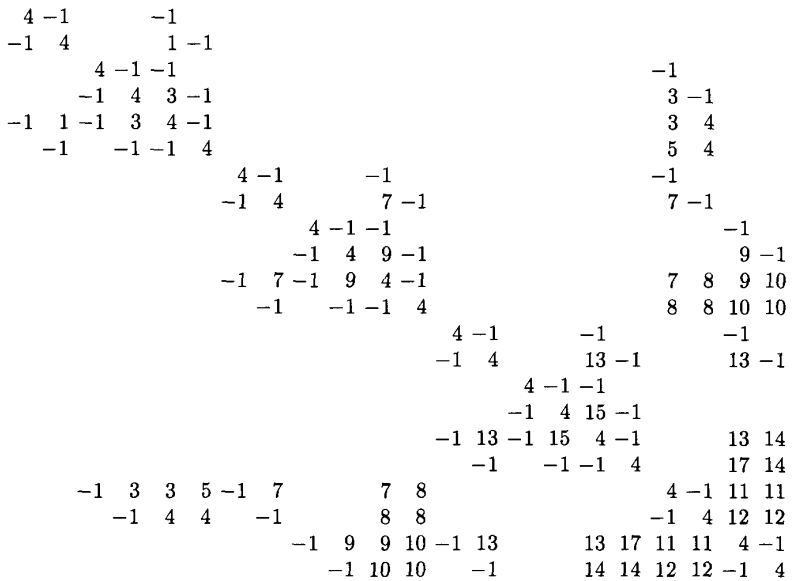

Figure 9.11: *Nested Dissection Matrix and Fill*

There are 60 fill elements in Figure 9.11, as opposed to 76 in the matrix of Figure 9.9, a relatively small but still significant saving. The main utility of these reordering techniques is, of course, for much larger problems.

In summary, the main purpose of this section has been to show that Gaussian elimination on the discrete equations of a partial differential equation leads to considerable fill, and hence extra computation, but by judicious reorderings of the equations the amount of fill can be significantly reduced.

Supplementary Discussion and References: 9.2

An excellent reference for further reading on direct methods for sparse linear systems is George and Liu [1981]. This book includes, in particular, detailed analyses of the storage and operation counts of one-way dissection and a more general treatment of nested dissection in which both horizontal and vertical separator lines are used. There also is a discussion and analysis of other reordering techniques that reduce the fill. For more general problems (i.e. not coming from Poisson's equation), one of the best such techniques is the *minimum degree algorithm*. Another good reference for sparse systems is

Duff et al. [1986].

The general idea of domain decomposition has broad application in the solution of partial differential equations, and has been a subject of intense research in the last several years because of its potential usefulness in parallel computing. For further discussion see, for example, Ortega [1988].

For general sparse linear systems, the algorithmic framework is the following:

Step 1. Use a reordering strategy to minimize fill.
Step 2. Do a symbolic factorization to determine the fill.
Set up data storage accordingly.
Step 3. Do the numerical factorization (LU or LL^T).
Step 4. Solve the corresponding triangular systems.

The symbolic factorization in Step 2 can be done surprisingly rapidly. Once this is done, the exact fill pattern is known so that the amount of storage needed for the factorization is also known. Storage can then be allowed for only the non-zero elements in the factors. Step 1 is predicated on the assumption that there is no need for interchanges to preserve numerical stability. If this is not the case, then interchanges for stability may conflict with interchanges to minimize fill. Generally, a compromise called *threshold pivoting* is used in which interchanges for stability are made only if the pivot element is too small, say less than .1 of the maximum element in its column. In this case there may be several candidates for the new pivot element, and the algorithm can choose the one that is best for maintaining sparsity.

For Poisson's equation and slight generalizations of it, there is another class of direct methods called *Fast Poisson Solvers.* There are many such methods and we will indicate only a few approaches. One is based on the fact that eigenvalues and eigenvectors of the matrix of (9.1.18) are known exactly in terms of sine functions. Thus if $A = QDQ^T$, where D is the diagonal matrix with the eigenvalues of A and Q is an orthogonal matrix whose columns are the eigenvectors of A, the solution of $A\mathbf{x} = \mathbf{b}$ is $\mathbf{x} = QD^{-1}Q^T\mathbf{b}$. The multiplications by Q^T and Q involve trigonometric sums and can be carried out by the *Fast Fourier Transform (FFT).* Other, still faster, methods are based on the *cyclic reduction* algorithm and a combination of this with the Fast Fourier Transform. For further discussion of Fast Poisson Solvers, see, for example, Hockney and Jesshope [1988].

EXERCISES 9.2

9.2.1. Let L and U be lower and upper triangular matrices for which $l_{ij} \neq 0$, $i \geq j$, and $u_{ij} \neq 0$, $i \leq j$. Show that all elements of UL are, in general, non-zero.

9.2.2. Consider the system

$$\begin{bmatrix} * & * & * & * & * \\ * & * & & & \\ * & & * & & \\ * & & & * & \\ * & & & & * \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix},$$

where $*$ denotes a non-zero element. Show that by the reordering of unknowns $x_1 \leftrightarrow x_5$, $x_2 \leftrightarrow x_4$, $x_3 \leftrightarrow x_3$ and the corresponding reordering of the equations, the system can be written as

$$\begin{bmatrix} * & & & & * \\ & * & & & * \\ & & * & & * \\ & & & * & * \\ * & * & * & * & * \end{bmatrix} \begin{bmatrix} x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \end{bmatrix}.$$

Generalize this to the case of the corresponding $n \times n$ system.

9.2.3. Show that the ordering of the grid points (and hence unknowns) of Figure 9.8 gives the coefficient matrix of 4's and -1's of Figure 9.9 for the system of equations (9.1.13). Next apply Gaussian elimination to this matrix and show that fill develops as indicated in Figure 9.9.

9.2.4. Show that the coefficient matrix A of Figure 9.9 (without the fill) is related to the matrix $\hat{A}$ that one would obtain from the natural ordering by $A = P\hat{A}P$, where P is a permutation matrix.

9.2.5. Verify that LU factorization of the matrix of (9.2.6) is given by (9.2.7) and (9.2.8).

9.2.6. For the grid of Figure 9.8, write out the 22×22 coefficient matrix for the natural ordering. Then verify, using the techniques that led to Figure 9.5, that the number of fill elements produced by Gaussian elimination is 182.

9.2.7. Consider the QR method for the matrix of Figure 9.5. Show that R fills in within the band.

9.2.8. Consider the special case of (9.2.6) in which

$$A = \begin{bmatrix} A_1 & B_1^T \\ B_1 & 0 \end{bmatrix},$$

and assume that A_1 is symmetric positive definite. Show that A is not positive definite but that there is a Cholesky-like factorization of the form

$$A = \begin{bmatrix} F & 0 \\ G & H \end{bmatrix} \begin{bmatrix} F^T & G^T \\ 0 & -H^T \end{bmatrix}.$$

Give an efficient algorithm for computing this factorization.

9.2.9. Consider the special case of Exercise 9.2.8 in which $A_1 = I$ and $B_1 = E^T$. Show that the solution of the normal equations $E^T E\mathbf{a} = E^T\mathbf{f}$ can be obtained by solving the system

$$\begin{bmatrix} I & E \\ E^T & 0 \end{bmatrix}\begin{bmatrix} \mathbf{r} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}.$$

Discuss the circumstances in which you might wish to solve this expanded system in place of the normal equations.

9.2.10. Use the fact that fill develops as in Figure 9.6 to show that the operation count for Gaussian elimination applied to the system (9.1.19) is $0(N^4)$.

9.3 Iterative Methods

An alternative to the direct methods discussed in the previous section is an iterative method, and we now describe some of the basic iterative methods for large sparse systems of equations.

Jacobi's Method

We consider the linear system $A\mathbf{x} = \mathbf{b}$ and make no assumptions about A at this time except that the diagonal elements are non-zero:

$$a_{ii} \neq 0, \qquad i = 1, \ldots, n. \tag{9.3.1}$$

Perhaps the simplest iterative procedure is *Jacobi's method*. Assume that an initial approximation $\mathbf{x}^0$ to the solution is chosen. Then the next iterate is given by

$$x_i^{(1)} = \frac{1}{a_{ii}}\Big(b_i - \sum_{j \neq i} a_{ij} x_j^{(0)}\Big), \qquad i = 1, \ldots, n. \tag{9.3.2}$$

It will be useful to write this in matrix-vector notation, and for this purpose, we let $D = \mathrm{diag}(a_{11}, \ldots, a_{nn})$ and $B = D - A$. Then it is easy to verify that (9.3.2) may be written as

$$\mathbf{x}^1 = D^{-1}(\mathbf{b} + B\mathbf{x}^0),$$

and the entire sequence of Jacobi iterates is defined by

$$\mathbf{x}^{k+1} = D^{-1}(\mathbf{b} + B\mathbf{x}^k), \qquad k = 0, 1, \ldots . \tag{9.3.3}$$

The Gauss-Seidel Method

A closely related iteration is derived from the following observation. After $x_1^{(1)}$ is computed in (9.3.2) it is available to use in the computation of $x_2^{(1)}$, and

it is natural to use this updated value rather than the original estimate $x_1^{(0)}$. If we use updated values as soon as they are available, then (9.3.2) becomes

$$x_i^{(1)} = \frac{1}{a_{ii}}\Big(b_i - \sum_{j<i} a_{ij}x_j^{(1)} - \sum_{j>i} a_{ij}x_j^{(0)}\Big), \qquad i = 1, \ldots, n, \tag{9.3.4}$$

which is the first step in the *Gauss-Seidel iteration.* To write this iteration in matrix-vector form, let $-L$ and $-U$ denote the strictly lower and upper triangular parts of A; that is, both L and U have zero main diagonals and

$$A = D - L - U. \tag{9.3.5}$$

If we multiply (9.3.4) through by a_{ii}, then it is easy to verify that the n equations in (9.3.4) can be written as

$$D\mathbf{x}^1 - L\mathbf{x}^1 = \mathbf{b} + U\mathbf{x}^0. \tag{9.3.6}$$

Since $D - L$ is a lower-triangular matrix with non-zero diagonal elements, it is nonsingular. Hence the entire sequence of Gauss-Seidel iterates is defined by

$$\mathbf{x}^{k+1} = (D - L)^{-1}[U\mathbf{x}^k + \mathbf{b}], \qquad k = 0, 1, \ldots\ . \tag{9.3.7}$$

The representations (9.3.3) and (9.3.7) of the Jacobi and Gauss-Seidel iterations are useful for theoretical purposes, but the actual computations would usually be done using the componentwise representations (9.3.2) and (9.3.4).

We next consider the application of these iterative methods to Laplace's equation on a square. The difference equations for this problem were given by (9.1.13) (with $f_{ij} = 0$) in the form

$$-u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} + 4u_{ij} = 0, \qquad i, j = 1, \ldots, N. \tag{9.3.8}$$

Here, the unknowns are the u_{ij}, $i, j = 1, \ldots, N$, and the remaining values of the u's are assumed known from the boundary conditions. Given initial approximations $u_{ij}^{(0)}$, a Jacobi step applied to (9.3.8) is

$$u_{ij}^{(1)} = \tfrac{1}{4}(u_{i+1,j}^{(0)} + u_{i-1,j}^{(0)} + u_{i,j+1}^{(0)} + u_{i,j-1}^{(0)}),$$

so that the new Jacobi approximation at the (i, j) grid point is simply the average of the previous approximations at the four surrounding grid points $(i \pm 1, j)$, $(i, j \pm 1)$. It is for this reason that the Jacobi method is sometimes known as the *method of simultaneous displacements.* Note that for the Jacobi method the order in which the equations are processed is immaterial. For the Gauss-Seidel method, this is not true, and each different ordering of the equations actually corresponds to a different iterative process. If we order the

grid points left to right and bottom to top, as was done in Section 9.1, then a typical Gauss-Seidel step is

$$u_{ij}^{(1)} = \tfrac{1}{4}(u_{i-1,j}^{(1)} + u_{i,j-1}^{(1)} + u_{i+1,j}^{(0)} + u_{i,j+1}^{(0)}).$$

This new approximation at the (i,j) grid point is again an average of the approximations at the four surrounding grid points, but now using two old values and two new values. The difference between the two methods is shown schematically in Figure 9.12.

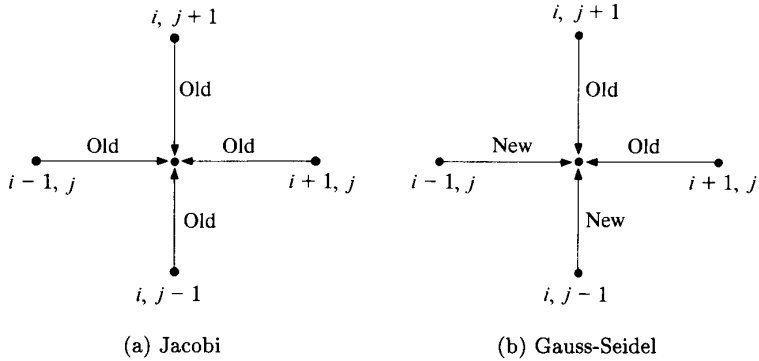

Figure 9.12: *Jacobi and Gauss-Seidel Updates*

Convergence

We consider next the question of the convergence of iterative methods. Both the Jacobi and Gauss-Seidel methods can be written in the form

$$\mathbf{x}^{k+1} = H\mathbf{x}^k + \mathbf{d}, \qquad k = 0, 1, \ldots . \tag{9.3.9}$$

In particular, $H = D^{-1}B$ and $\mathbf{d} = D^{-1}\mathbf{b}$ for the Jacobi process, whereas $H = (D-L)^{-1}U$ and $\mathbf{d} = (D-L)^{-1}\mathbf{b}$ for Gauss-Seidel. Now assume that $\mathbf{x}^*$ is the exact solution of the system $A\mathbf{x} = \mathbf{b}$. For the Jacobi method we then have

$$(D-B)\mathbf{x}^* = \mathbf{b} \quad \text{or} \quad \mathbf{x}^* = D^{-1}B\mathbf{x}^* + D^{-1}\mathbf{b},$$

and for the Gauss-Seidel method

$$(D-L-U)\mathbf{x}^* = \mathbf{b} \quad \text{or} \quad \mathbf{x}^* = (D-L)^{-1}U\mathbf{x}^* + (D-L)^{-1}\mathbf{b}.$$

Thus in both cases

$$\mathbf{x}^* = H\mathbf{x}^* + \mathbf{d}. \tag{9.3.10}$$

If we subtract (9.3.10) from (9.3.9), we have

$$\mathbf{e}^{k+1} = H\mathbf{e}^k, \qquad k = 0, 1, \ldots, \tag{9.3.11}$$

where $\mathbf{e}^k = \mathbf{x}^k - \mathbf{x}^*$ is the error at the kth step.

Iterative methods of the form (9.3.9) are called *stationary one-step* methods and (9.3.10) is the *consistency condition.* Then (9.3.11) is the basic error relation for such methods. We can analyze the errors in much the same way as we analyzed the power method in Section 7.3. Assume that H has n linearly independent eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$. The initial error $\mathbf{e}^0$ can then be expressed as some linear combination of the eigenvectors:

$$\mathbf{e}^0 = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_n\mathbf{v}_n. \tag{9.3.12}$$

Thus,

$$\mathbf{e}^k = H^k\mathbf{e}^0 = c_1\lambda_1^k\mathbf{v}_1 + c_2\lambda_2^k\mathbf{v}_2 + \cdots + c_n\lambda_n^k\mathbf{v}_n. \tag{9.3.13}$$

In order that $\mathbf{e}^k \to 0$ as $k \to \infty$ for any $\mathbf{x}^0$ (and, hence, any c_i in (9.3.12)), we must have $|\lambda_i| < 1$, $i = 1, \ldots, n$; that is, the spectral radius, $\rho(H)$, must be less than one. This result is true also when H does not have n linearly independent eigenvectors, but the proof is more difficult (see the Supplementary Discussion). We state this basic convergence theorem as:

THEOREM 9.3.1 *If (9.3.10) holds, the iterates* (9.3.9) *converge to the solution* $\mathbf{x}^*$ *for any starting vector* $\mathbf{x}^0$ *if and only if* $\rho(H) < 1$.

Theorem 9.3.1 is the basic theoretical result for one-step iterative methods but it does not immediately tell us if a particular iterative method is convergent; we need to ascertain if the spectral radius of the iteration matrix for the method is less than 1. In general, this is a very difficult problem, for which one might have to resort to computing all the eigenvalues of the iteration matrix. But for some iterative methods and for certain classes of matrices it is relatively easy to determine that the convergence criterion is satisfied. We next give some examples of this for the Jacobi and Gauss-Seidel methods.

THEOREM 9.3.2 *Assume that the matrix A is strictly diagonally dominant:*

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \qquad i = 1, \ldots, n. \tag{9.3.14}$$

Then both the Jacobi and Gauss-Seidel iterations converge to the unique solution of $A\mathbf{x} = \mathbf{b}$ *for any starting vector* $\mathbf{x}^0$.

The proof of this theorem is very simple for the Jacobi method. Since $H = D^{-1}B$, the condition (9.3.14) implies that the sums of the absolute values of the elements in each row of H are less that 1. Hence $||H||_\infty < 1$, and therefore all eigenvalues of H are less that 1 in absolute value. Thus Theorem 9.3.1 applies. The proof for Gauss-Seidel is a little more complicated. Let λ be any eigenvalue of H and $\mathbf{v}$ a corresponding eigenvector. Then

$$\lambda \mathbf{v} = H\mathbf{v} = (D - L)^{-1}U\mathbf{v},$$

or

$$\lambda(D - L)\mathbf{v} = U\mathbf{v}. \tag{9.3.15}$$

Let

$$|v_k| = \max\{|v_i| \ : \ i = 1, \ldots, n\}. \tag{9.3.16}$$

The kth equation of (9.3.15) is

$$\lambda(a_{kk}v_k + \sum_{j<k} a_{kj}v_j) = -\sum_{j>k} a_{kj}v_j, \tag{9.3.17}$$

and we set

$$\alpha = \sum_{j<k} \frac{a_{kj}v_j}{a_{kk}v_k}, \qquad \beta = \sum_{j>k} \frac{a_{kj}v_j}{a_{kk}v_k}.$$

Then (9.3.17) can be written as

$$\lambda(1 + \alpha) = -\beta,$$

so that

$$|\lambda| \leq \frac{|\beta|}{|1 + \alpha|} \leq \frac{|\beta|}{1 - |\alpha|} < 1,$$

by (9.3.14) and (9.3.16). Thus we have shown that $\rho(H) < 1$ and Theorem 9.3.1 applies.

The condition of strict diagonal dominance is a rather stringent one and does not apply to the difference equations (9.3.8) for Laplace's equation: in most rows of the coefficient matrix there are four coefficients of absolute value 1 in the off-diagonal positions, so that strict inequality does not hold in (9.3.14). However, by using different techniques (see the Supplementary Discussion), it can be shown that both methods indeed converge for the difference equations (9.3.8).

The coefficient matrix of the equations (9.3.8) is symmetric [see (9.1.18)], and it can be shown that it is also positive-definite. Indeed, for many discrete analogs of elliptic partial differential equations, the coefficient matrix will be symmetric and positive-definite. In this case the Gauss-Seidel iteration will always converge, although symmetry and positive-definiteness is not, in general, sufficient for the Jacobi method to converge. We state the following theorem without proof:

THEOREM 9.3.3 *Assume that the matrix A is symmetric and positive-definite. Then the Gauss-Seidel iterates converge to the unique solution of $A\mathbf{x} = \mathbf{b}$ for any starting vector $\mathbf{x}^0$.*

Even when the Jacobi and Gauss-Seidel methods are convergent, the rate of convergence may be so slow as to preclude their usefulness; this is particularly so for discrete analogs of elliptic partial differential equations. For example, for equation (9.3.8) with $N = 44$, the error in each iteration of the Gauss-Seidel method will decrease asymptotically only by a factor of about 0.995. Moreover, the Jacobi method is about twice as slow on this problem, and the rate of convergence of both methods becomes worse as N increases.

The SOR Method

In certain cases it is possible to accelerate considerably the rate of convergence of the Gauss-Seidel method. Given the current approximation $\mathbf{x}^k$, we first compute the Gauss-Seidel iterate

$$\hat{x}_i^{(k+1)} = \frac{1}{a_{ii}}\Big(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\Big) \tag{9.3.18}$$

as an intermediate value, and then take the final value of the new approximation to the ith component to be

$$x_i^{(k+1)} = x_i^{(k)} + \omega(\hat{x}_i^{(k+1)} - x_i^{(k)}). \tag{9.3.19}$$

Here ω is a parameter that has been introduced to accelerate the rate of convergence.

We can rewrite (9.3.18) and (9.3.19) in the following way. First substitute (9.3.18) into (9.3.19):

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\Big(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\Big) \tag{9.3.20}$$

and then rearrange the equation into the form

$$a_{ii}x_i^{(k+1)} + \omega\sum_{j<i} a_{ij}x_i^{(k+1)} = (1-\omega)a_{ii}x_i^{(k)} - \omega\sum_{j>i} a_{ij}x_j^{(k)} + \omega b_i.$$

This relationship between the new iterates $x_i^{(k+1)}$ and the old $x_i^{(k)}$ holds for $i = 1, \ldots, n$, and using (9.3.5) we can write it in matrix-vector form as

$$D\mathbf{x}^{k+1} - \omega L\mathbf{x}^{k+1} = (1-\omega)D\mathbf{x}^k + \omega U\mathbf{x}^k + \omega\mathbf{b}.$$

Since the matrix $D - \omega L$ is again lower-triangular and, by assumption, has non-zero diagonal elements, it is nonsingular, so we may write

$$x^{k+1} = (D - \omega L)^{-1}[(1-\omega)D + \omega U]\mathbf{x}^k + \omega(D - \omega L)^{-1}\mathbf{b}. \tag{9.3.21}$$

This defines the *successive overrelaxation (SOR) method*, although, as with Gauss-Seidel, the componentwise prescription (9.3.18), (9.3.19) would usuallly be used for the actual computation. Note that if $\omega = 1$, (9.3.21) reduces to the Gauss-Seidel iteration.

We restrict ourselves to real values of the parameter ω. Then a necessary condition for the *SOR* iteration (9.3.21) to be convergent is that $0 < \omega < 2$ (see Exercise 9.3.15). In general, a choice of ω in this range will *not* give convergence, but in the important case that the coefficient matrix A is symmetric and positive-definite, we have the following extension of Theorem 9.3.3, which we also state without proof:

> THEOREM 9.3.4 (Ostrowski) *Assume that A is symmetric and positive-definite. Then for any $\omega \in (0, 2)$ and any starting vector $\mathbf{x}^0$, the SOR iterates* (9.3.21) *converge to the solution of $A\mathbf{x} = \mathbf{b}$.*

We would like to be able to choose the parameter ω so as to optimize the rate of convergence of the iteration (9.3.21). In general this is a very difficult problem, and we will attempt to summarize, without proofs, a few of the things that are known about its solution. For a class of matrices that are called *consistently ordered with property A*, there is a rather complete theory that relates the rate of convergence of the SOR method to that of the Jacobi method and gives important insights into how to choose the optimum value of ω. We will not define this class of matrices precisely; suffice it to say that it includes the matrix (9.1.18) of equations (9.3.8) as well as many other matrices that arise as discrete analogs of elliptic partial differential equations.

The fundamental result that holds for this class of matrices is a relationship between the eigenvalues of the SOR iteration matrix

$$H_\omega = (D - \omega L)^{-1}[(1 - \omega)D + \omega U] \tag{9.3.22}$$

and the eigenvalues μ_i of the Jacobi iteration matrix $J = D^{-1}(L + U)$. Under the assumption that the μ_i are all real and less that 1 in absolute value, it can be shown that the optimum value of ω, denoted by ω_0, is given in terms of the spectral radius, $\rho(J)$, of J by

$$\omega_0 = \frac{2}{1 + \sqrt{1 - \rho^2}}, \qquad \rho = \rho(J), \tag{9.3.23}$$

and is always between 1 and 2. The corresponding value of the spectral radius of H_ω is

$$\rho(H_{\omega_0}) = \omega_0 - 1, \tag{9.3.24}$$

and it is this spectral radius that governs the ultimate rate of convergence of the method. Moreover, we can ascertain the behavior of $\rho(H_\omega)$ as a function of ω, as is shown in Figure 9.13.

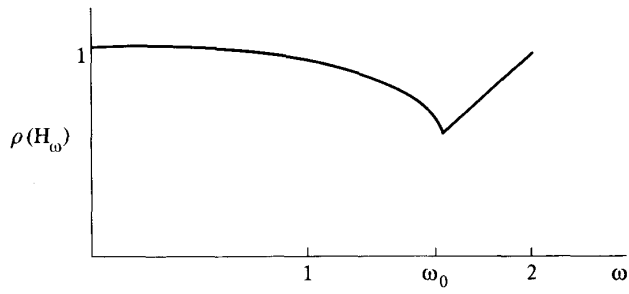

Figure 9.13: $\rho(H_\omega)$ *as a Function of* ω

We can obtain an idea of the acceleration of convergence that is possible by considering the equations (9.3.8). For this problem the eigenvalues of the Jacobi iteration matrix J can be computed explicitly, and the largest turns out to be

$$\rho(J) = \cos \pi h, \qquad h = \frac{1}{N+1}. \tag{9.3.25}$$

If we put this in (9.3.23), we obtain

$$\omega_0 = \frac{2}{1 + \sqrt{1 - \cos^2 \pi h}}, \qquad \rho(H_{\omega_0}) = \frac{1 - \sqrt{1 - \cos^2 \pi h}}{1 + \sqrt{1 - \cos^2 \pi h}}. \tag{9.3.26}$$

If, again for illustration, we take $N = 44$, then

$$p(J) \doteq 0.9976, \quad \rho(H_1) \doteq 0.995, \quad \omega_0 \doteq 1.87, \quad \rho(H_{\omega_0}) \doteq 0.87. \tag{9.3.27}$$

This shows that, asymptotically, the error in Jacobi's method will decrease by a factor of 0.9976 at each step, and that of the Gauss-Seidel method by a factor of $0.995 = (.9976)^2$, which is twice as fast. But the error in the SOR method will decrease by a factor of $0.87 = (.995)^{30}$, so that SOR is about thirty times as fast as the Gauss-Seidel method. Moreover, the improvement becomes more marked as N increases (see Exercise 9.3.8).

The preceding discussion indicates that dramatic improvements in the rate of convergence of the Gauss-Seidel method are possible. However, a number of caveats are in order. First of all, many – perhaps most – large sparse matrices that arise in practice do not enjoy being "consistently ordered with property A," and the preceding theory will not hold. It is still possible that introduction of the parameter ω into the Gauss-Seidel method will produce a substantial increase in the rate of convergence, but this will not necessarily be known in advance, nor will we know how to choose a good value of ω. Even if the coefficient matrix is "consistently ordered with property A," it

still may be difficult to obtain a good estimate of ω_0. It was possible to compute explicitly the quantities of (9.3.27) only because of the very special nature of the equations (9.3.8), which allowed an exact computation of $\rho(J)$. In general this will not be possible, and to use (9.3.26) will require estimating $\rho(J)$, which is itself a difficult problem. Thus even in those cases where the preceding theory holds, it may be necessary to use an approximation process to obtain a suitable value of ω. In particular, there are "adaptive methods" that help to approximate a good value of ω as the SOR iteration proceeds (see the Supplementary Discussion).

The Conjugate Gradient Method

A large number of iterative methods for solving linear systems of equations can be derived as minimization methods. If A is symmetric and positive definite, then the quadratic function

$$q(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b} \tag{9.3.28}$$

has a unique minimizer which is the solution of $A\mathbf{x} = \mathbf{b}$ (Exercise 9.3.11). Thus methods that attempt to minimize (9.3.28) are also methods to solve $A\mathbf{x} = \mathbf{b}$. Many minimization methods for (9.3.28) can be written in the form

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \mathbf{p}^k, \qquad k = 0, 1, \ldots . \tag{9.3.29}$$

Given the *direction vector* $\mathbf{p}^k$, one way to choose α_k is to minimize q along the line $\mathbf{x}^k - \alpha\mathbf{p}^k$; that is,

$$q(\mathbf{x}^k - \alpha_k\mathbf{p}^k) = \min_{\alpha} q(\mathbf{x}^k - \alpha\mathbf{p}^k). \tag{9.3.30}$$

For fixed $\mathbf{x}^k$ and $\mathbf{p}^k$, $q(\mathbf{x}^k - \alpha\mathbf{p}^k)$ is a quadratic function of α and may be minimized explicitly (Exercise 9.3.12) to give

$$\alpha_k = -(\mathbf{p}^k, \mathbf{r}^k)/(\mathbf{p}^k, A\mathbf{p}^k), \qquad \mathbf{r}^k = \mathbf{b} - A\mathbf{x}^k. \tag{9.3.31}$$

In (9.3.31), and henceforth, we use the notation $(\mathbf{u}, \mathbf{v})$ to denote the inner product $\mathbf{u}^T\mathbf{v}$.

Although there are many other ways to choose the α_k, we will use only (9.3.31) and concentrate on different choices of the direction vectors $\mathbf{p}^k$. One simple choice is $\mathbf{p}^k = \mathbf{r}^k$, which gives the *method of steepest descent*:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k(\mathbf{b} - A\mathbf{x}^k), \qquad k = 0, 1, \ldots . \tag{9.3.32}$$

This is also known as *Richardson's method* and is closely related to Jacobi's method (Exercise 9.3.13). As with Jacobi's method, the convergence of (9.3.32) is usually very slow. Another simple strategy is to take $\mathbf{p}^k$ as one of the unit vectors $\mathbf{e}_i$, which has a 1 in position i and is zero elsewhere. Then, if

$\mathbf{p}^0 = \mathbf{e}_1, \mathbf{p}^1 = \mathbf{e}_2, \ldots, \mathbf{p}^{n-1} = \mathbf{e}_n$, and the α_k are chosen by (9.3.31), n steps of (9.3.29) are equivalent to one Gauss-Seidel iteration on the system $A\mathbf{x} = \mathbf{b}$ (Exercise 9.3.14).

A very interesting choice of direction vectors arises from requiring that they satisfy

$$(\mathbf{p}^i, A\mathbf{p}^j) = 0, \qquad i \neq j. \tag{9.3.33}$$

Such vectors are called *conjugate* (with respect to A). It can be shown that if $\mathbf{p}^0, \ldots, \mathbf{p}^{n-1}$ are conjugate and the α_k are chosen by (9.3.31), then the iterates $\mathbf{x}^k$ of (9.3.29) converge to the exact solution in at most n steps. This property, however, is not useful in practice because of rounding error; moreover, for large problems n is far too many iterations. For many problems, however, especially those arising from partial differential equations, methods based on conjugate directions may converge, up to a convergence criterion, in far fewer than n steps.

To use a conjugate direction method it is necessary to obtain the vectors $\mathbf{p}^k$ that satisfy (9.3.33). The *preconditioned conjugate gradient* algorithm generates these vectors as part of the overall method:

Choose $\mathbf{x}^0$. Set $\mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0$. Solve $M\tilde{\mathbf{r}}^0 = \mathbf{r}^0$. Set $\mathbf{p}^0 = \tilde{\mathbf{r}}^0$. (9.3.34a)

For $k = 0, 1, \ldots$

$$\alpha_k = -(\tilde{\mathbf{r}}^k, \mathbf{r}^k)/(\mathbf{p}^k, A\mathbf{p}^k) \tag{9.3.34b}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \mathbf{p}^k \tag{9.3.34c}$$

$$\mathbf{r}^{k+1} = \mathbf{r}^k + \alpha_k A\mathbf{p}^k \tag{9.3.34d}$$

Test for convergence (9.3.34e)

Solve $M\tilde{\mathbf{r}}^{k+1} = \mathbf{r}^{k+1}$ (9.3.34f)

$$\beta_k = (\tilde{\mathbf{r}}^{k+1}, \mathbf{r}^{k+1})/(\tilde{\mathbf{r}}^k, \mathbf{r}^k) \tag{9.3.34g}$$

$$\mathbf{p}^{k+1} = \tilde{\mathbf{r}}^{k+1} + \beta_k \mathbf{p}^k \tag{9.3.34h}$$

We next make several comments about this algorithm. Assume for now that $M = I$ in (9.3.34a,f); then $\tilde{\mathbf{r}}^k = \mathbf{r}^k$ and (9.3.34) defines the *conjugate gradiant method.* The formula for α_k in (9.3.34b) differs from (9.3.31) because of an identity, $(\mathbf{p}^k, \mathbf{r}^k) = (\mathbf{r}^k, \mathbf{r}^k)$, in the conjugate gradient method; thus (9.3.34b,c) compute the minimum in the direction $\mathbf{p}^k$. The next step computes the new residual vector since

$$\mathbf{r}^{k+1} = \mathbf{b} - A\mathbf{x}^{k+1} = b - A(\mathbf{x}^k - \alpha_k \mathbf{p}^k) = \mathbf{r}^k + \alpha A\mathbf{p}^k.$$

Because $A\mathbf{p}^k$ is already known, this way of computing $\mathbf{r}^{k+1}$ saves the multiplication $A\mathbf{x}^{k+1}$. Next there would be a convergence test; the usual one is $(\mathbf{r}^{k+1}, \mathbf{r}^{k+1}) = ||\mathbf{r}^{k+1}||_2^2 \leq \varepsilon$. Finally, the last two steps, (9.3.34g,h), compute a new direction vector $\mathbf{p}^{k+1}$. It is not obvious that the direction vectors computed in this fashion are conjugate, but it can be proved relatively easily (Exercise 9.3.9).

Assuming still that $M = I$, the major work in each step of the conjugate gradient algorithm is the matrix-vector product $A\mathbf{p}^k$. Note that as in the Lanczos algorithm of Chapter 7, A need not be known explicitly as long as a matrix-vector product can be computed. In addition to this product, three "saxpy" operations of the form vector + scalar * vector are required, as well as inner products for $(\mathbf{p}^k, A\mathbf{p}^k)$ and $(\mathbf{r}^k, \mathbf{r}^k)$; if the latter quantity is used in the convergence test, no further work is required there except a comparison.

We now return to the assumption that $M = I$. It can be shown that if $\mathbf{x}^*$ is the exact solution, then

$$||\mathbf{x}^k - \mathbf{x}^*||_2 \leq 2\sqrt{\kappa}\alpha^k ||x^0 - x^*||_2, \tag{9.3.35a}$$

where $\kappa = \text{cond}_2(A) = ||A||_2 ||A^{-1}||_2$ and

$$\alpha = (\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1). \tag{9.3.35b}$$

The larger the condition number, the closer α is to 1 and the slower the rate of convergence. The role of the matrix M in (9.3.34f) is to "precondition" the matrix A and reduce its condition number so as to obtain faster convergence.

Preconditioning

There are numerous ways to choose the matrix M but we discuss only two. First, consider $M = D$, the main diagonal of A. This can be very beneficial if the main diagonal elements of A have considerable variability. On the other hand, if they are constant, as for the Poisson equations (9.1.13), then this choice of M does not change the rate of convergence. Another, more generally applicable, way to choose M is known as *incomplete Cholesky* factorization:

For $j = 1, \ldots, n$

$$l_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 \right)^{1/2} \tag{9.3.36a}$$

For $i = j + 1, \ldots, n$

If $a_{ij} = 0$ then $l_{ij} = 0$ else (9.3.36b)

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}}{l_{jj}}. \tag{9.3.36c}$$

Without the statement (9.3.36b), this is the Cholesky factorization of Figure 4.5. The effect of (9.3.36b) is to bypass the computation of l_{ij} if the corresponding element of A is zero. Thus (9.3.36b) suppresses the fill that would normally occur in a Cholesky factorization, and we obtain a "no-fill" approximate Cholesky factor L. We then take $M = LL^T$. The incomplete factorization (9.3.36) is done once at the beginning of the iteration; then during each iteration (9.3.34f) is carried out by solving two triangular systems with coefficient matrices L and L^T. The hope is that the extra work of incorporating this preconditioner will be more than repaid by a reduction in the iteration count. For example, for a two dimensional Laplace equation on a 99×101 grid (9999 equations), the conjugate gradient iteration takes 266 iterations (for a given convergence criterion), whereas with incomplete Cholesky preconditioning this is reduced to 45 iterations. The extra work to incorporate the preconditioning does no more than double the time per iteration (and can be much less with certain optimizations) so that the gain in speed is at least a factor of three. Although the Cholesky factorization can always be performed if A is symmetric positive definite, the incomplete factorization of (9.3.36) may fail. However, it is possible to modify (9.3.36) in various ways so that an incomplete factorization is possible (see the Supplementary Discussion).

Summary

In this section we have given an introduction to some of the simplest iterative methods for large sparse systems of linear equations, especially those which arise from elliptic partial differential equations. There are a number of other methods, some of which are mentioned in the Supplementary Discussion, and which method to use on a given problem is usually not clear. Moreover, direct methods such as Gaussian elimination with suitable reorderings are sometimes very efficient. In fact, for elliptic equations in two dimensions, direct methods are probably to be preferred, whereas for three-dimensional problems iterative methods are probably the best. However, which method to use will depend upon several factors, including the computer, the particular equation to be solved, and the accuracy required in the solution.

Supplementary Discussion and References: 9.3

The Jacobi and Gauss-Seidel iterations are classical methods that go back to the last century. The basic theory of the SOR method was developed by Young [1950]. For a complete discussion of the Jacobi, Gauss-Seidel, and SOR methods and their many variants, see Varga [1962] and Young [1971]. For ways to compute adaptively the ω in SOR, see Hageman and Young [1981] and Young and Mai [1990].

The basic convergence Theorem 9.3.1 is really equivalent to powers of a matrix tending to zero. The condition that $H^k \mathbf{e}_0 \to 0$ as $k \to \infty$ for any $\mathbf{e}_0$ is equivalent to $H^k \to 0$ as $k \to \infty$. If $H = PJP^{-1}$, where J is the Jordan

canonical form of H, then $H^k = PJ^kP^{-1}$ and $H^k \to 0$ if and only if $J^k \to 0$ as $k \to \infty$. If J is diagonal, then the result is essentially what was shown in the text. Otherwise one needs to analyze powers of a Jordan block. By considering $(\lambda I + E)^k$, where E is a matrix of 1's on the first superdiagonal, it is easy to see that for $k \geq n$

$$\begin{bmatrix} \lambda & 1 & & \\ & \lambda & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{bmatrix}^k = \begin{bmatrix} \lambda^k & k\lambda^{k-1} & \binom{k}{2}\lambda^{k-2} & \cdots & \binom{k}{n-1}\lambda^{k-n+1} \\ & \lambda^k & k\lambda^{k-1} & \cdots & \vdots \\ & & \ddots & \ddots & \\ & & & & k\lambda^{k-1} \\ & & & & \lambda^k \end{bmatrix},$$

which shows that the powers of a Jordan block tend to zero if and only if $|\lambda| < 1$. Although $\rho(H) < 1$ is a necessary and sufficient condition that the iterates (9.3.9) converge for any $\mathbf{x}^0$, it is not necessarily the case that the relation $||\mathbf{e}^{k+1}|| < ||\mathbf{e}^k||$ for the error vectors will hold in some usual norm (see Exercise 9.3.10).

Theorem 9.3.2 can be extended to a form suitable for handling equations such as (9.3.8). If A is irreducibly diagonally dominant (see the Supplementary Discussion of Section 3.1), then the system of equations $A\mathbf{x} = \mathbf{b}$ has a unique solution $\mathbf{x}^*$, and both the Jacobi and Gauss-Seidel iterates converge to $\mathbf{x}^*$ for any starting vector $\mathbf{x}^0$. It can be shown that the coefficient matrix of (9.3.8) is irreducible and that this result then applies.

Theorem 9.3.4 is a special case of a more general convergence theorem: if A is symmetric positive definite and $A = P - Q$, where P is nonsingular and $\mathbf{x}^T(P + Q)\mathbf{x} > 0$ for all $\mathbf{x} \neq 0$, then $\rho(P^{-1}Q) < 1$. This is sometimes referred to as the Householder-John Theorem but it is actually due to Weissinger; see Ortega [1990], where it is called the P-regular splitting theorem. A converse of Theorem 9.3.4 also holds: If A is symmetric with positive diagonal elements and SOR converges for some $\omega \in (0, 2)$ and every $\mathbf{x}^0$, then A is positive definite. If this converse is added, Theorem 9.3.4 is known as the Ostrowski-Reich Theorem.

Iterative methods such as Gauss-Seidel tend to make fairly rapid progress in the early stages and then slow down. As a consequence Gauss-Seidel is rarely used by itself but it can be a key component of the *multigrid* method. Assume that a partial differential equation is discretized on a grid of points for which the approximate solution is desired. We call this the *fine* grid, and subsets of these grid points are *coarser* grids. A multigrid method takes a few Gauss-Seidel iterations on the fine grid, then stops and "restricts" information from the fine grid to a coarser grid and performs a few Gauss-Seidel iterations on this coarser grid. The process is continued by using still coarser grids. Information from these coarser grids is transmitted back to the finer grids by interpolation. There

are many important details in this process and many variations on the basic multigrid idea. Properly implemented, multigrid methods are increasingly becoming the method of choice for many problems. For an introduction to the multigrid method, see Briggs [1987] and, for a more advanced treatment, Hackbrush [1985].

The conjugate gradient method was introduced by Hestenes and Stiefel [1952]. Because of its finite convergence property it is a direct method, although it was not a successful competitor against Gaussian elimination. Since the 1970's, however, it has enjoyed a resurgence as an iterative method for large sparse systems. Another sometimes useful way to view the conjugate gradient method is as an acceleration technique. It can be shown that the conjugate gradient iterates satisfy the three-term recurrence relation known as the *Rutishauser form*,

$$\mathbf{x}^{k+1} = \rho_{k+1}\{\delta_{k+1}[(I - A)\mathbf{x}^k + \mathbf{b}] + (1 - \delta_{k+1})\mathbf{x}^k\} + (1 - \rho_{k+1})\mathbf{x}^{k-1}, \quad (9.3.37)$$

where the scalar parameters ρ_k and δ_k are themselves computed by recurrence relations. In this form, (9.3.37) can be considered an acceleration of the basic iterative method $\mathbf{y}^{k+1} = (I - A)\mathbf{y}^k + \mathbf{b}$, which is (9.3.32) with the $\alpha_k = 1$. For further reading on the conjugate gradient method, see Golub and Van Loan [1989], Ortega [1988], and Hageman and Young [1981]. In particular, see Ortega [1988] for various ways to modify the incomplete Cholesky factorization (9.3.36) in case it should fail.

Many elliptic equations that arise in practice are nonlinear. The methods of this section do not apply immediately in this case, although extensions of them to nonlinear equations have been developed (see Ortega and Rheinboldt [1970]). On the other hand, if a method such as Newton's method is used (Section 5.3), then at each stage a large sparse linear system must be solved approximately, and iterative methods can be used for this purpose.

EXERCISES 9.3

9.3.1. Apply the Jacobi and Gauss-Seidel iterations to the system of equations $A\mathbf{x} = \mathbf{b}$ where

$$A = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

Use the starting approximation $\mathbf{x}^0 = (1, 1, 1)$ and take enough steps of the iterative processes for the pattern of convergence to become clear.

9.3.2. Write computer programs for the Jacobi and Gauss-Seidel methods. Test them on the problem of Exercise 9.3.1.

9.3.3. Write out in detail the Jacobi and Gauss-Seidel iterations for the equations (9.3.8) for $N = 3$.

9.3.4. Consider the elliptic equation $u_{xx} + u_{yy} + cu = 0$ with the values of u prescribed on the boundary of a square domain. Derive the difference equations corresponding to (9.3.8). If c is a negative constant, show that the resulting coefficient matrix is strictly diagonally dominant.

9.3.5. Let A be a real $n \times n$ symmetric positive definite matrix.

a. Show that the diagonal elements of A are necessarily positive.

b. If C is any real $n \times n$ nonsingular matrix, show that $C^T AC$ is also positive definite.

9.3.6. Carry out several steps of the SOR iteration for the problem of Exercise 9.3.1. Use the values $\omega = 0.6$ and $\omega = 1.4$ and compare the rates of convergence to the Gauss-Seidel iteration.

9.3.7. Write a computer program to apply the SOR iteration to (9.3.8).

9.3.8. Use the relations (9.3.25) and (9.3.26) to compute $\rho(J)$, ω_0, and $\rho(H_{\omega_0})$ for (9.3.8) for $N = 99$ and $N = 999$.

9.3.9. Prove that the relationship (9.3.33) holds for the conjugate gradient method by using the following induction argument. As the induction hypothesis, assume that $(\mathbf{p}^k, A\mathbf{p}^j) = (\mathbf{r}^k, \mathbf{r}^j) = 0$, $j = 0, \ldots, k-1$. ($(\mathbf{p}^1, A\mathbf{p}^0) = 0$; $(\mathbf{r}^1, \mathbf{r}^0) = (\mathbf{r}^0 - \alpha_0 A\mathbf{p}^0, \mathbf{p}^0) = 0$ by the definition of α_0.) Then show that $(\mathbf{p}^{k+1}, A\mathbf{p}^j) = (\mathbf{r}^{k+1}, \mathbf{r}^j) = 0$ for $j = 0, 1, \ldots, k$.

9.3.10. Let $H = \begin{bmatrix} 0.5 & \alpha \\ 0 & 0 \end{bmatrix}$. Compute H^k and show that if $\mathbf{e}^0 = (0,1)^T$, then $\mathbf{e}^k = H^k\mathbf{e}^0 = (\alpha 2^{-k+1}, 0)^T$. Thus if $\alpha = 2^0$, $||e^k||_2 \geq ||e_0||_2$.

9.3.11. Show that the gradient vector for (9.3.28) is $\nabla q(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$. If A is symmetric positive definite, show that the unique solution of $A\mathbf{x} = \mathbf{b}$ is the unique minimizer of $\mathbf{q}$.

9.3.12. For the function q of (9.3.28), show that

$$q(\mathbf{x} - \alpha\mathbf{p}) = \tfrac{1}{2}\mathbf{q}^T A\mathbf{p}\alpha^2 + \mathbf{p}^T(\mathbf{b} - A\mathbf{x})\alpha - \tfrac{1}{2}\mathbf{x}^T(2\mathbf{b} - A\mathbf{x}).$$

For $\mathbf{x} = \mathbf{x}^k$ and $\mathbf{p} = \mathbf{p}^k$, minimize this function of α to obtain (9.3.31).

9.3.13. If $\alpha_k \equiv 1$ and A is such that its main diagonal is I, show that (9.3.32) is Jacobi's method.

9.3.14. Let $\mathbf{p}^0 = \mathbf{e}_1, \mathbf{p}^1 = \mathbf{e}_2, \ldots, \mathbf{p}^{n-1} = \mathbf{e}_n$, where $\mathbf{e}_i$ is the vector with 1 in the ith component and zero elsewhere. Show that n steps of (9.3.29) with α_k given by (9.3.31) is equivalent to one Gauss-Seidel iteration on the system $A\mathbf{x} = \mathbf{b}$.

9.3.15. For the *SOR* iteration matrix (9.3.22), show that $\det(D - \omega L)^{-1} = \det D^{-1}$ and then that $\det H_\omega = (1-\omega)^n$. Using the fact that the determinant of a matrix is the product of its eigenvalues, conclude that if ω is real and $\omega \leq 0$ or $\omega \geq 2$, then at least one eigenvalue of H_ω must be greater than or equal to 1 in magnitude.

9.3.16. Let $A = \epsilon L + D - U$, where ϵ is a small parameter and $\|U\|_\infty = 1$.

a. Give an upper bound on the spectral radius of the Gauss-Seidel iteration matrix $(\epsilon L + D)^{-1}U$.

b. Now consider the "backward" Gauss-Seidel iteration

$$(D + U)\mathbf{x}^{k+1} = -\epsilon L\mathbf{x}^k + \mathbf{b}.$$

Give an upper bound on the spectral radius of the iteration matrix for this method and show its dependence on ϵ.