Chapter 13

Boundary Value Problems for Partial Differential Equations*

Elliptic equations constitute the third category of partial differential equations. As a prototype, we take the *Poisson equation*

$$\begin{aligned}\nabla^2 u(x,\, y) &= \frac{\partial^2 u(x,\, y)}{\partial\, x^2} + \frac{\partial^2 u(x,\, y)}{\partial\, y^2} \\ &= g(x,\, y),\end{aligned} \tag{13.1}$$

which is to be satisfied by the function $u(x,\ y)$ in some region Ω of the xy plane, with the right side $g(x,\ y)$ given on all of Ω. The special case $g(x,\ y) = 0$ is *Laplace's* equation and the differential operator ∇^2 is called the *Laplacian*. Elliptic partial differential equations occur frequently as models for static situations in potential theory, elasticity, and thermal conduction. There is no time variable involved, so the approximation has to be determined for all points at once.

The subsidiary conditions for elliptic equations are entirely boundary conditions. We assume that Ω is a finite two-dimensional region whose boundary is denoted by Γ. Boundary conditions will be specified on Γ. We assume that the boundary is a single closed curve and that it is smooth in the sense that ν, the outward normal to the boundary curve, is a continuously differentiable function along Γ (Figure 13.1).

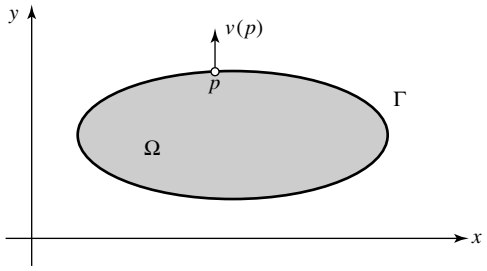


Figure 13.1
Region Ω and its boundary Γ.

We will limit our discussion to a particularly simple type of boundary condition in which we specify that the value of u on the boundary by

$$u(x, y) = \alpha(x, y), \quad \text{for all } (x, y) \in \Gamma. \tag{13.2}$$

In partial differential equations literature these are known as *Dirichlet* boundary conditions.

13.1 Finite Difference Methods

The construction of finite difference methods for the Poisson equation is quite intuitive. We first construct a rectangular grid on Ω. Here we will use a grid with uniform spacing h in both the x and y directions, but this can be easily changed. We then use finite differences to approximate the Laplacian. When a grid point has all of its four neighbors in Ω, we say that it is a *regular* point and use the *five-point star* arrangement shown in Figure 13.2, which leads to the approximation

$$\begin{aligned} &\nabla^2 u(x_i, y_j) \\ &\quad \cong \frac{u(x_{i-1}, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 4u(x_i, y_j)}{h^2}. \end{aligned} \tag{13.3}$$

At most interior points we can use this standard approximation for the Laplacian, but some changes may have to be made near the boundaries. When the boundary is curved the grid lines will intersect the boundary at some nongrid points and we get *irregular* interior points. The distance of

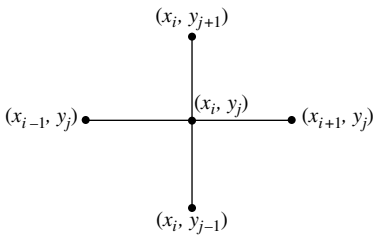


Figure 13.2
Regular five-point star pattern.

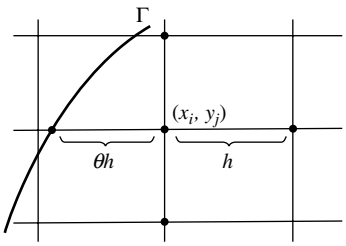


Figure 13.3
Grid spacing at an irregular point.

these irregular interior points from the boundary points is less than the grid spacing, so the formulas for approximating the derivatives have to be adjusted. For the situation shown in Figure 13.3, we find from the discussion of finite difference approximations that

$$\frac{\partial^2 u(x_i,\, y_j)}{\partial\, x^2} \cong \frac{2\theta u(x_{i+1},\, y_j) + 2\alpha(x_i - \theta h,\, y_j) - 2(1+\theta)u(x_i,\, y_j)}{\theta(1+\theta)h^2} \tag{13.4}$$

is an approximation of the second partial with respect to x at the grid point $(x_i,\ y_j)$. Using this or similar formulas for the derivative in y, we can derive a suitable approximation for the Laplacian at other irregular points.

The finite difference approximations are then used to replace the derivatives in equation (13.1), and the equation is satisfied at all the interior points. If we use $U_{i,\,j}$ to denote the approximation to $u(x_i,\ y_j)$, we get equations of the form

$$\frac{U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{i,j}}{h^2} = g(x_i,\, y_j) \tag{13.5}$$

at all regular points, with suitably modified formulas for the irregular points. In any case, if there are m interior points we will have exactly m equations and m unknowns, so in principle the system can be solved by standard matrix methods.

The double index scheme in the grid labeling is convenient for visualization, but to make the equations (13.5) into a standard matrix problem we have to translate this into a single index scheme. We do this by sequentially labeling interior points as $P_1,\ P_2,\ \ldots,\ P_m$ and the boundary points as $B_1,\ B_2,\ \ldots,\ B_k$. The unknowns in (13.5) are converted into a vector $\mathbf{U}$, such that $[\mathbf{U}]_i$ represents the approximation corresponding to the grid point P_i; that is,

$$[\mathbf{U}]_i = u(P_i) = U_{k,l},$$

where $(x_k,\ y_l)$ are the coordinates of the point P_i. The set of equations can then be written in matrix form as

$$\mathbf{AU} = \mathbf{y}, \tag{13.6}$$

where the elements of the matrix $\mathbf{A}$ and the right side $\mathbf{y}$ depend on the particular arrangement of the grid and the boundary.

The specific form of the matrix $\mathbf{A}$ depends on the way we go from the double index to the single index notation. Often one uses a systematic numbering scheme in which we proceed along one column of grid points, say bottom to top. When we reach the top of one row, we begin again at the bottom of the adjacent column. This *natural* ordering, illustrated in the next example, is often used.

Example 13.1 Suppose that we want to solve Poisson's equation in the interior of an ellipse Ω with

$$\Gamma = \{(x, y)\colon\ -1 \le x \le 1,\ x^2 + 1.44y^2 = 1\}.$$

Choosing $h = 0.4$, we get the grid shown in Figure 13.4. The labeling of the points is column-wise, as suggested. There are a total of 16 interior points, so that $\mathbf{A}$ will be a 16×16 matrix. The point P_6 is a regular point, with neighbors $P_2,\ P_5,\ P_7$, and P_{10}. The sixth row of $\mathbf{A}$ will therefore have the elements

$$[\mathbf{A}]_{66} = -\frac{4}{h^2},$$
$$[\mathbf{A}]_{62} = [\mathbf{A}]_{65} = [\mathbf{A}]_{67} = [\mathbf{A}]_{6,10} = \frac{1}{h^2}.$$

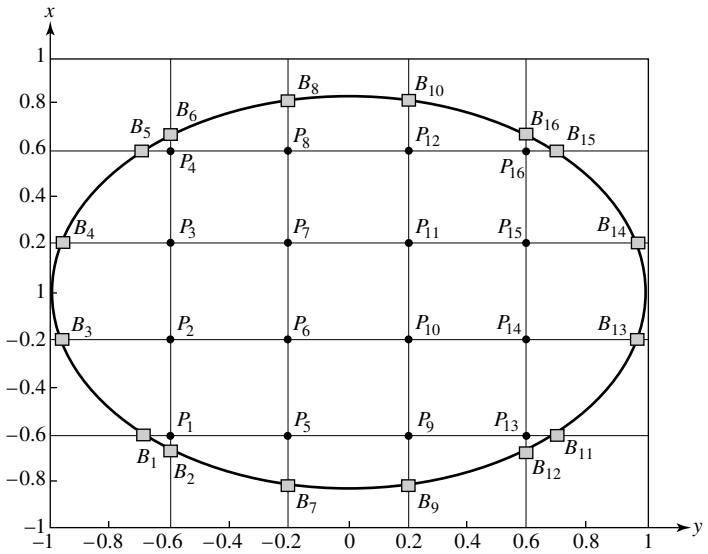


Figure 13.4 Grid and boundary points ordering for Example 13.1.

All other entries in this row will be zero. The right side has the component

$$[\mathbf{y}]_6 = g(P_6).$$

The point P_8 is irregular, so the boundary comes into play. For differentiation in the y direction we need to use the appropriately modified formula (13.4),

$$\frac{\partial^2 u(P_8)}{\partial y^2} \cong \frac{2\theta\alpha(B_8) + 2u(P_7) - 2(1+\theta)u(P_8)}{\theta(1+\theta)h^2}.$$

The y coordinate of B_8 is 0.8165, so that $\theta = 0.5414$. Replacing u by its approximation $\mathbf{U}$ and using the boundary value, this gives

$$\frac{\partial^2 u(P_8)}{\partial y^2} \cong \frac{1.083\alpha(B_8) + 2[\mathbf{U}]_7 - 3.083[\mathbf{U}]_8}{0.817h^2}.$$

Putting this together with the centered difference approximation in the x direction, we find that the nonzero elements in the eighth row of $\mathbf{A}$ are

$$[\mathbf{A}]_{8,8} = -\frac{5.695}{h^2},$$

$$[\mathbf{A}]_{8,4} = [\mathbf{A}]_{8,12} = \frac{1}{h^2},$$

$$[\mathbf{A}]_{8,7} = \frac{1.300}{h^2}.$$

The right side has

$$[\mathbf{y}]_8 = g(P_8) - \frac{2.398\alpha(B_8)}{h^2}.$$

All the other elements of $\mathbf{A}$ and $\mathbf{y}$ can be found in a similar way. ■

This example illustrates an aspect that complicates the task of solving partial differential equations. The curved boundaries affect the construction of the matrix $\mathbf{A}$ and make it a nontrivial problem. For more complicated equations with difficult boundaries, constructing the approximating equations is a major source of concern and writing a computer program that does it is a lengthy undertaking. Creating software for realistic partial differential equations usually involves serious data management problems and rarely is an easy task.

There are two other major issues that we have to consider. We need to produce an error analysis that tells us how the method can be expected to work, and we need to find efficient methods for solving the large systems that we get.

EXERCISES

1. Derive the approximation (13.4).
2. Show how (13.3) needs to be modified when the grid spacing in the x and y directions differs.
3. In Example 13.1, find the elements of the second row of $\mathbf{A}$.
4. In Example 13.1, find the pattern of the nonzero elements of $\mathbf{A}$.
5. Suppose that Ω is the unit square $0 \leq x \leq 1,\ 0 \leq y \leq 1$. Describe what the coefficient matrix $\mathbf{A}$ looks like in this case if the natural ordering illustrated in Example 13.1 is used.
6. How does the discussion in this section have to be changed for the solution of the equation

$$\frac{\partial^2 u(x,\, y)}{\partial\, x^2} + \frac{\partial^2 u(x,\, y)}{\partial\, y^2} + u(x,\, y) \ = g(x,\, y)?$$

13.2 Error Analysis for the Finite Difference Method

To analyze the error in this finite difference method, we proceed as we did in the analysis of the two-point boundary value problem. We use

$$\mathbf{u} = \begin{bmatrix} u(P_1) \\ u(P_2) \\ \vdots \\ u(P_m) \end{bmatrix}$$

to denote the vector of true solutions at the grid points, and consider the error vector $\mathbf{e} = (e_1,\ e_2,\ \ldots,\ e_m)^T$ with

$$e_i = u(P_i) - [\mathbf{U}]_i.$$

We are interested in the behavior of $||\mathbf{e}||$ as h decreases.

The local discretization error $\mathbf{t}(u,\ h) = (\tau_1,\ \tau_2,\ \ldots,\ \tau_m)^T$ is defined as

$$\tau_i = \nabla^2 u(P_i) - [\mathbf{Au}]_i \tag{13.7}$$

and our goal is to establish the connection between the local discretization error and the error in the approximate solution. Noting that

$$\begin{aligned} [\mathbf{AU}]_i &= g(P_i) \\ &= \nabla^2 u(P_i), \end{aligned}$$

we get

$$[\mathbf{A}(\mathbf{U}-\mathbf{u})]_i = \tau_i,$$

or

$$\mathbf{Ae} = -\mathbf{t}(u,h).$$

If $\mathbf{A}$ is invertible, then

$$||\mathbf{e}|| \leq ||\mathbf{A}^{-1}||\ ||\mathbf{t}(u,h)\,||. \tag{13.8}$$

The local discretization error is usually quite tractable. If assumptions on the smoothness of u can be made, it is only a matter of bounding the error in numerical differentiation. The question of the convergence of the finite difference then becomes a matter of bounding $||\mathbf{A}^{-1}||$ as a function of h. This is the stability question which is, as usual, a difficult issue. For finite difference methods for partial differential equations this often means appealing to special results from matrix theory. For the case under discussion, the arguments can be found in various places (see for example Isaacson and Keller [5], p.447) and it is known that $||\mathbf{A}^{-1}||_\infty$ is bounded as $h \to 0$. Accepting this, we can conclude that the finite difference method described converges and that the error is proportional to the local discretization error. We can also conclude that the condition number of (13.5),

$$\begin{aligned} k &= ||\mathbf{A}||_\infty\ ||\mathbf{A}^{-1}||_\infty \\ &= O(\frac{1}{h^2}), \end{aligned}$$

so there will be some limited accumulation of rounding error.

Exercises

1. Suppose that Ω is the unit square and that u is four times continuously differentiable in both variables. Show that then $||\mathbf{t}(u,\ h)||_\infty = O(h^2)$.
2. Show that the discretized Laplacian on the unit square satisfies a *maximum* principle that guarantees that the maximum grid value must be located on the boundary. Use this to demonstrate that the numerical method is stable. (Hint: Assume that the maximum value occurs at an interior point and show that this leads to a contradiction.)
3. How is the bound in Exercise 1 affected by curved boundaries?
4. How is the stability argument in Exercise 2 affected by curved boundaries?
5. Construct the matrix $\mathbf{A}$ for a unit square and empirically examine $||\,\mathbf{A}^{-1}\,||$ as a function of h.

13.3 Solving Finite Difference Systems

Since the number of unknowns in the discretized system $m = O(1/h^2)$, even a moderately small h leads to a large number of equations. For complicated regions, or when high accuracy is required, the system (13.5) may contain in excess of 10,000 equations. This makes it impractical to use a normal GEM algorithm. Fortunately, the matrix $\mathbf{A}$ in (13.6) has special properties that we can take advantage of.

As we know, direct methods for sparse matrices are potentially effective as long as we can control the fill-in. For banded matrices we can do this to some extent. For rectangular regions, or even for simple convex regions such as that in Example 13.1, the structure of the matrix is predictable. For a rectangle and a natural ordering, the matrix will be banded with bandwidth roughly

$$\beta \cong 2\sqrt{m},$$

as shown in Figure 13.5. If we apply the Gauss elimination method restricted to the band, we find that the reduction of each column requires about β^2 multiplications and additions so that, if no row interchanges are necessary, the entire triangularization takes

$$N \cong 4m^2$$

operations. For $m = 10^4$, this is within the range of present-day computers, but still takes a great deal of time.

The band reduction methods are not as efficient as possible because they do not take advantage of the large number of zeros within the band. In recent years there has been a great deal of interest in faster direct methods. Many algorithms have been developed that can solve large problems quite well, but this is a special topic that is not easily treated in an elementary fashion. Details can be found in books devoted to this subject, for example, Duff, Erisman, and Reid [8].

Although direct methods have many advantages, the simplicity of iterative methods often makes them attractive alternatives. Historically, it-

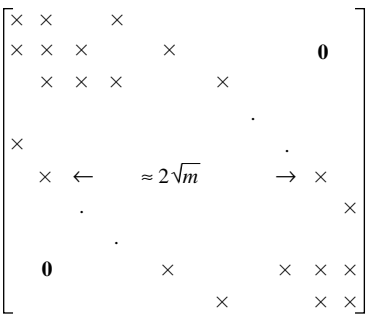


Figure 13.5
Band structure for the finite difference method on a rectangle.

erative methods for finite difference systems were the first to be used, and their performance has been studied for many years. A large number of useful results are known, of which we will present only the simplest. To get an intuitive grasp for how the iterations are performed, it is helpful to switch back to the two-dimensional indexing scheme for the unknowns, in which $U_{i,\,j}$ is an approximation to $u(x_i,\ y_j)$. With this convention the equation at a regular grid point can be written as

$$U_{i,j} = \frac{1}{4}\Big((U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1}) + h^2 g(x_i, y_j)\Big), \tag{13.9}$$

with suitable modifications for the irregular grid points. If we let $U_{i,\,j}^{[k]}$ denote the kth iterate for $U_{i,\,j}$, then (13.9) suggests the simple iteration

$$U_{i,j}^{[k+1]} = \frac{1}{4}\left(U_{i-1,j}^{[k]} + U_{i+1,j}^{[k]} + U_{i,j-1}^{[k]} + U_{i,j+1}^{[k]} + h^2 g(x_i, y_j)\right). \tag{13.10}$$

This is known as the *Jacobi iteration*.

If we think of a cycle of the Jacobi method as one iteration on all unknowns, we see that in one cycle only the values of the previous cycle are used. The entire array is updated when the cycle has been completed, hence the result of one cycle does not depend on the order in which the new iterates are computed. We can change this and use the new iterates as soon as they are available, but in this case, the order in which the points are taken does matter. For the ordering scheme in Figure 13.4, the iteration can be defined by

$$U_{i,j}^{[k+1]} = \frac{1}{4}\left(U_{i-1,j}^{[k+1]} + U_{i+1,j}^{[k]} + U_{i,j-1}^{[k+1]} + U_{i,j+1}^{[k]} + h^2 g(x_i, y_j)\right). \tag{13.11}$$

If we compute the unknowns at the points P_i in the order $i = 1,\ 2,\ \ldots$, then the quantities on the right of (13.11) are known whenever they are needed, and so the formula is a straightforward successive substitution. The algorithm is called the *Gauss-Seidel* method.

To analyze the convergence of these schemes, we return to vector notation and write

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{R},$$

where $\mathbf{D}$ is a diagonal matrix, $\mathbf{L}$ is lower triangular, and $\mathbf{R}$ is upper triangular, both of the latter with zero diagonals. The Jacobi method can then be written as

$$\mathbf{U}^{[k+1]} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{R})\mathbf{U}^{[k]} + \mathbf{D}^{-1}\mathbf{y}, \tag{13.12}$$

while the Gauss-Seidel method is

$$\mathbf{U}^{[k+1]} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{R}\mathbf{U}^{[k]} + (\mathbf{D} + \mathbf{L})^{-1}\mathbf{y}. \tag{13.13}$$

We know from the study of iterative methods that a sufficient condition for the convergence of successive substitutions

$$\mathbf{U}^{[k+1]} = \mathbf{B}\mathbf{U}^{[k]} + \mathbf{y}$$

is that

$$||\mathbf{B}|| < 1.$$

For symmetric matrices and the 2-norm, this is equivalent to the bound on the spectral radius,

$$\rho(\mathbf{B}) = r < 1. \tag{13.14}$$

While $\mathbf{B}$ is not always symmetric, it can be shown that this condition is also sufficient for nonsymmetric matrices. Therefore, the speed of convergence is governed by the magnitude of r, with the error in the kth iteration proportional to r^k. The convergence question is then reduced to finding the dominant eigenvalues of the matrices involved in the iteration.

For special cases, such as rectangular Ω where all grid points are regular, one can find the eigenvalues explicitly, but generally this is a difficult task. We want to therefore appeal to general results in spectral theory that allow us to establish (13.14). We cannot use Gershgorin's theorem directly because the necessary conditions are not satisfied. For example, in Jacobi's method, where

$$\mathbf{B} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{R}),$$

we find that for all i

$$\mathbf{B}_{ii} = 0,$$

and

$$\sum_{j \neq i} |\,[\mathbf{B}]_{i,j}| \leq 1, \tag{13.15}$$

with equality for some rows. All we can conclude from this is that

$$r \leq 1,$$

which is not sufficient to establish convergence. Fortunately, there are some rows of $\mathbf{B}$ for which strict inequality in (13.15) is attained. This gives us the hope that condition (13.14) could hold. A detailed analysis shows that this is indeed so, and that for Jacobi's method

$$r = 1 - |\eta|, \tag{13.16}$$

where $\eta = O(h^2)$. A similar result holds for the Gauss-Seidel method, with η about twice the magnitude of that for Jacobi's method. From this we conclude that both iterations converge, with better results to be expected from the Gauss-Seidel method.

However, (13.16) also shows that r is quite close to one, and suggests that convergence will be slow and deteriorate with decreasing h. While each iteration can be carried out quickly, usually many iterations have to be done before the desired accuracy is achieved. This has spurred on the study of alternatives that accelerate the convergence. One of a number of methods that will do this is the *over-relaxation* method, which is an instance of extrapolated iteration. If we can estimate the contraction factor r, the extrapolated value can be used as the new iterate, leading to the over-relaxation formula

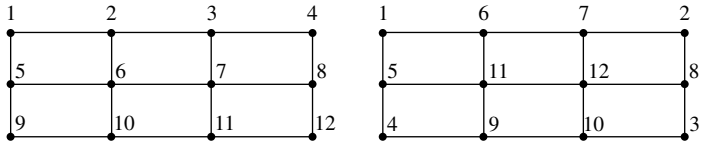
$$\mathbf{U}^{[k+1]} = \frac{1}{1-r}\left\{-(\mathbf{D}+\mathbf{L})^{-1}\mathbf{R}\mathbf{U}^{[k]} + (\mathbf{D}+\mathbf{L})^{-1}\mathbf{y}\right\} - \frac{r}{1-r}\mathbf{U}^{[k]}. \quad (13.17)$$

If we have a good guess for r, this can improve the speed of convergence considerably.

The topic of iterative methods for these systems is an extensive and complicated subject that is explored in the relevant literature. Varga's book [25] is a classic in this area.

Exercises

1. To solve a finite difference system on a three by four grid, which of the two ordering schemes below would be preferable?



2. Suppose that Ω is a rectangular region twice as long in the x-direction as in the y-direction. If we use a finite difference discretization in which $\Delta x = \Delta y$ and a natural ordering for the grid points, is there any advantage of ordering by columns as opposed to by rows?

3. Verify that (13.13) is the correct matrix form of the Gauss-Seidel method.

4. Approximately how many more Jacobi iterations than Gauss-Seidel iterations are needed to reduce the error by two orders of magnitude?

5. Show that even in the presence of curved boundaries (13.15) holds and that strict inequality is attained for some rows.

13.4 The Finite Element Method

The finite element method is a popular approach to the numerical solution of elliptic partial differential equations. Much has been written about it in numerical analysis literature and it has also been widely used and discussed by structural engineers. We can characterize the finite element method as a Galerkin-type method, with local, often very simple bases, and some manipulations to reduce the smoothness needed for the basis of the approximations. While our discussion will focus on Poisson's equation, much of what we say is applicable to more general elliptic partial differential equations.

Galerkin's method involves writing the approximation as

$$u_n(x, y) = \sum_{i=1}^{n} c_i \phi_i(x, y), \tag{13.18}$$

and then selecting the expansion coefficients so that the residual of the approximation is orthogonal to all the expansion functions; that is, that

$$\int_\Omega \left\{ \nabla^2 u_n(x, y) - g(x, y) \right\} \phi_i(x, y) dx dy = 0 \tag{13.19}$$

for all $i = 1,\ 2,\ \ldots,\ n$.

In addition to the orthogonality constraints, we also need to address the boundary conditions. For simplicity we will assume that the Dirichlet conditions are homogeneous, so the boundary values in (13.2) are

$$\alpha(x, y) = 0. \tag{13.20}$$

If we choose the basis functions ϕ_i so that each of them is zero on the boundary, the linear combination (13.18) satisfies (13.20). As we will see, this is generally not hard to do.

Writing the orthogonality condition in the form (13.19) apparently implies that the approximation, and hence every basis function, is twice continuously differentiable in both variables. But such high continuity requirements are often inconvenient and, if possible, we try to reduce them. The trick is integration by parts, using *Green's theorem*. One of the forms of Green's theorem is as follows: If Ω is a closed and bounded region in the plane, and if u and v are both twice continuously differentiable functions that vanish on the boundary Γ, then

$$\int_\Omega \left(\frac{\partial^2 u}{\partial s^2} + \frac{\partial^2 u}{\partial t^2} \right) v d\Omega = -\int_\Omega \left(\frac{\partial u}{\partial s}\frac{\partial v}{\partial s} + \frac{\partial u}{\partial t}\frac{\partial v}{\partial t} \right) d\Omega. \tag{13.21}$$

Applying this to the Laplacian and the expansion functions in (13.18), we have

$$\int_\Omega \left(\frac{\partial^2 u_n}{\partial x^2} + \frac{\partial^2 u_n}{\partial y^2} \right) \phi_i d\Omega = -\int_\Omega \left(\frac{\partial u_n}{\partial x}\frac{\partial \phi_i}{\partial x} + \frac{\partial u_n}{\partial y}\frac{\partial \phi_i}{\partial y} \right) d\Omega. \tag{13.22}$$

The right side not only replaces second derivatives by first order differentiation, but remains sensible as long as the first partial derivatives of all the functions are integrable. It can be shown that extending the approximation in this way does not change the solution, so we will use the right side of (13.22) instead of the left one. The implication is that we can use basis functions that are piecewise differentiable, with possibly bounded jump discontinuities in the first derivatives.

With this modification, Galerkin's method then reduces to a linear system

$$\mathbf{A}\mathbf{c} = \mathbf{y}, \tag{13.23}$$

where

$$[\mathbf{A}]_{ij} = -\int_\Omega \left(\frac{\partial \phi_i}{\partial x}\frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial y}\frac{\partial \phi_j}{\partial y} \right) d\Omega, \tag{13.24}$$

and

$$[\mathbf{y}]_i = \int_\Omega g\phi_i d\Omega. \tag{13.25}$$

In the finite element literature the matrix $\mathbf{A}$ is called the *stiffness matrix*. Choosing the basis elements so that the homogeneous boundary conditions (13.20) are satisfied, and so that the stiffness matrix can be computed without too much difficulty, is the primary challenge in the practical use of the finite element method.

One of the simplest and most popular choices is a piecewise planar approximation on triangles. Suppose for the moment that Ω is a polygonal region. Then it can be *triangulated*, meaning that it can be partitioned into a set of triangles (Figure 13.6).

The approximation u_n should be a continuous, planar function on each triangle. This can be achieved by using basis elements which have the value one at one vertex P of a triangle and zero at all other vertices, that are planar in all triangles with P as a vertex (Figure 13.7), and that are zero everywhere else. A linear combination of such basis functions will then have the required continuity properties. We also note that if we use basis

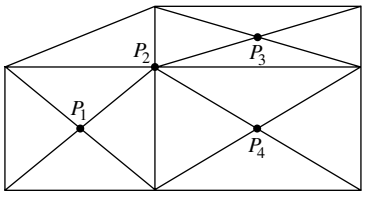


Figure 13.6
A triangulation of a polygonal region.

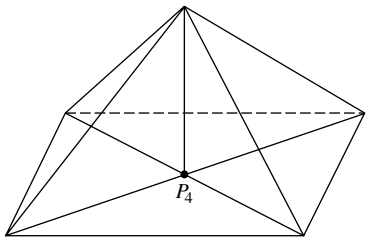


Figure 13.7
A basis element for piecewise planar approximations.

functions only for the interior points, the homogeneous boundary conditions (13.18) are automatically satisfied for any linear combination. For such piecewise planar elements, the matrix entries $[\mathbf{A}]_{ij}$ are easily computed analytically. The computations of the elements in (13.25) may have to be done numerically, but normally this does not cause any great difficulty.

We rarely know of an obvious and efficient triangulation that will solve a problem to a specified accuracy. Usually, we start with some rough triangulation, then subdivide those triangles that are likely to give the best improvement. Some care needs to be taken when subdividing a triangle. If the subdivision tends to make the triangles skinnier, the accuracy will suffer. A way of avoiding this is shown in Figure 13.8, where a triangle is divided into four smaller ones, each similar to the original one. In this way, we can adaptively place smaller triangles into regions where the solution changes rapidly and use larger triangles where the solution is very smooth. Re-computing the stiffness matrix after subdivision is relatively straightforward. Unlike the finite difference method, having triangles of widely varying size does not affect the method significantly. This is a strong advantage for the finite element method.

Flexibility is one of the main attractive features of the finite element method. Not only is subdividing a region easy, the triangular elements are very suitable for complex geometry. In addition, the method rests on strong physical and mathematical foundations, making it suitable for automatic implementation. There are also some disadvantages. Writing a computer program that finds a good triangulation can be challenging. The stiffness matrix is more complicated than the corresponding matrix for the finite

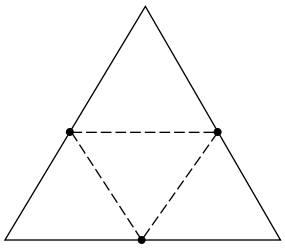


Figure 13.8
Subdividing a triangle, using the midpoints of each side.

difference methods. Although generally sparse, the sparsity pattern in the stiffness matrix is not as predictable as it is for the finite difference case. All of this makes it quite difficult to implement the finite element method. But most of these difficulties have been overcome. For many important elliptic partial differential equations there exist programs that produce good triangulations, construct the stiffness matrix, and solve the resulting sparse linear system efficiently. The widespread availability of such software is a major factor in the popularity of the finite element method.

Much has been written about the theory and the practice of finite element methods, their connection with variational principles, their convergence, and their stability. All of this is quite complicated, but there are a number of books that give a fairly accessible presentation. Good sources for further reading are Johnson [13] and Prenter [21].

Exercises

1. Suppose that in the Dirichlet problem the boundary conditions are not homogeneous. What can be done to make it suitable for a finite element treatment?

2. Find an explicit expression for

$$[\mathbf{A}]_{ij} = -\int_{\Omega} \left(\frac{\partial \phi_i}{\partial x}\frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial y}\frac{\partial \phi_j}{\partial y} \right) d\Omega,$$

where the ϕ_i are piecewise planar elements and Ω is a single triangle.

3. How does the triangulation affect the sparsity pattern of the stiffness matrix? What are typically the average number of elements in a row of the sparsity matrix?

4. Show that the subdivision suggested in Figure 13.8 produces four congruent triangles, each similar to the original triangle.

13.5 Multidimensional Equations

The partial differential equations that one encounters in practice are usually quite a bit more complicated than the simple prototypes we have studied here. Frequently, there are more than two dimensions involved. This introduces new complications coming from the geometry, increases the computer resources required, and often precludes any practically useful error analysis. Still, with the guidelines that we get from the simpler models, we can devise methods and successfully solve many very complicated equations.

As one step in the direction of more complicated models, we consider the *two-dimensional heat equation*

$$\frac{\partial u(x, y, t)}{\partial t} = \frac{\partial^2 u(x, y, t)}{\partial x^2} + \frac{\partial^2 u(x, y, t)}{\partial y^2}, \tag{13.26}$$

in some spatial region Ω with boundary Γ, in a time interval $0 \le t \le T$. There are initial conditions

$$u(x, y, 0) = g_1(x, y),$$

for all $(x, y) \in \Omega$, and boundary conditions

$$u(x, y, t) = g_2(x, y, t)$$

for all $(x, y) \in \Gamma$.

Equation (13.26) is a parabolic equation. It is an initial value problem that behaves very much like the one-dimensional heat equation (12.1). We can therefore model our numerical methods on the one-dimensional case. But in the steady-state case

$$\frac{\partial u(x, y, t)}{\partial t} = 0,$$

(13.26) reduces to Laplace's equation, so we should expect some aspects of boundary value problems to become significant.

Extending the framework of Section 12.1, we introduce a two-dimensional space grid, defined by grid points (x_i, y_j), $i = 0, 1, \ldots, n_1$, $j = 0, 1, \ldots, n_2$. The grid will be assumed to be uniform in each direction, with respective grid spacing Δx and Δy. A time grid is defined by time lines $t_0, t_1, \ldots$, spaced at a distance Δt. The approximation to $u(x_i, y_j, t_k)$ will be denoted by $U_{i,j,k}$. To construct a finite difference method, we replace the Laplacian by its approximation on a five-point star (Figure 13.2) and the time derivative by a difference. As before, the question is what time difference to use.

The analog of the explicit method (12.10) is obvious; we simply use a forward time difference to get

$$\begin{aligned} U_{i,j,k+1} = U_{i,j,k} &+ \Delta t \frac{U_{i-1,j,k} + U_{i+1,j,k} - 2U_{i,j,k}}{(\Delta x)^2} \\ &+ \Delta t \frac{U_{i,j-1,k} + U_{i,j+1,k} - 2U_{i,j,k}}{(\Delta y)^2} \end{aligned} \tag{13.27}$$

for regular interior points, with suitable changes for the points near the boundary.

With the explicit method the solution can be computed easily with a forward marching process. We need to be concerned about stability and, not surprisingly, it turns out that the method is stable only for the condition

$$\frac{\Delta t}{(\Delta x)^2 + (\Delta y)^2} \leq \frac{1}{2}. \tag{13.28}$$

The one-dimensional case suggests that we can alleviate the stability condition and increase the accuracy by going to an implicit scheme, such as an analog of the Crank-Nicholson method (12.14). Unfortunately, when we do this we find that we have to solve a complete Laplace equation at each time-step. This is quite expensive, so we need to look for cheaper alternatives.

One successful strategy is the *alternating direction implicit* (ADI) method. Each time-step is split into two equal parts. In the first part, integrating from $k\Delta t$ to $(k + \frac{1}{2})\Delta t$, x is taken to be implicit while y is treated explicitly. This gives

$$\begin{aligned} U_{i,j,k+1/2} = U_{i,j,k} &+ \frac{\Delta t}{2(\Delta x)^2}\left(U_{i-1,j,k+1/2} + U_{i+1,j,k+1/2} - 2U_{i,j,k+1/2}\right) \\ &+ \frac{\Delta t}{2(\Delta y)^2}\left(U_{i,j-1,k} + U_{i,j+1,k} - 2U_{i,j,k}\right). \end{aligned} \tag{13.29}$$

In the next step, from $(k + \frac{1}{2})\Delta t$ to $(k + 1)\Delta t$, the method is made explicit in x and implicit in y, so that

$$\begin{aligned} U_{i,j,k+1} = U_{i,j,k+1/2} &+ \frac{\Delta t}{2(\Delta x)^2}\left(U_{i-1,j,k+1/2} + U_{i+1,j,k+1/2} - 2U_{i,j,k+1/2}\right) \\ &+ \frac{\Delta t}{2(\Delta y)^2}\left(U_{i,j-1,k+1} + U_{i,j+1,k+1} - 2U_{i,j,k+1}\right). \end{aligned} \tag{13.30}$$

Each time step now involves solving a number of implicit equations, but all of them are essentially tridiagonal. This is much cheaper than solving a complete Laplace equation at each step. The ADI method is known to be stable for all Δx, Δy, and Δt.