6.1. Introduction.

So far in our exposition we have focused primarily on numerical methods for solving first- or second-order linear PDEs. While this class of equations is important and quite broad, in many problems of interest in science and engineering the appropriate PDEs may fall outside its boundaries. For example, some PDEs arising in structural mechanics involve derivatives of order higher than two. The biharmonic equation developed in Section 1.4 is one of the important paradigms in this regard. Also, many of the PDEs occurring in applications are nonlinear. While we have devoted some attention to nonlinear hyperbolic equations in Chapter Five, we have yet to examine the numerical treatment of nonlinear PDEs of elliptic or parabolic type. Finally, a plethora of applications require the solution of simultaneous, coupled PDEs. These applications are so numerous that we cannot hope to give a systematic treatment of them all here. We shall, however, give an overview of two important coupled systems, namely, the problem of a deforming solid and the problem of simultaneous flow of oil, gas, and water in a petroleum reservoir.

6.2. The Biharmonic Equation.

Let us begin by considering the biharmonic equation,

$$\nabla^4 u = 0, \tag{6.2-1}$$

on a two-dimensional domain Ω, where $\nabla^4 = \partial^4/\partial x^4 + 2\partial^4/\partial x^2 \partial y^2 + \partial^4/\partial y^4$ in Cartesian coordinates. As Section 1.4 explains, this equation governs the Airy stress function for plane strain in an elastic solid. The equation arises in related contexts as well. For example, it governs the transverse deflection of a two-dimensional elastic plate under a transverse

load. Two sets of boundary conditions commonly occur for this equation. The first set is

$$\begin{aligned} u(\mathbf{x}) &= \alpha(\mathbf{x}) \ , \\ \nabla^2 u(\mathbf{x}) &= \beta(\mathbf{x}) \ , \end{aligned} \tag{6.2-2}$$

for $\mathbf{x} \in \partial\Omega$. When $\alpha(\mathbf{x}) \equiv \beta(\mathbf{x}) \equiv 0$ in flat-plate applications, these boundary conditions model a plate with simply supported edges. The second set is

$$u(\mathbf{x}) = \frac{\partial u}{\partial n}(\mathbf{x}) = 0 \ , \quad \mathbf{x} \in \partial\Omega \ , \tag{6.2-3}$$

where $\partial u/\partial n = \nabla u \cdot \mathbf{n}$ stands for the outward normal derivative of u on $\partial\Omega$. These boundary conditions represent a plate with clamped edges.

There is a noteworthy difference between boundary conditions (6.2-2) and (6.2-3). When we impose the simply-supported plate conditions (6.2-2), the boundary-value problem for Equation (6.2-1) admits a factored form

$$\begin{aligned} \nabla^2 u(\mathbf{x}) &= v(\mathbf{x}) \ , & \mathbf{x} &\in \Omega \ , \\ u(\mathbf{x}) &= \alpha(\mathbf{x}) \ , & \mathbf{x} &\in \partial\Omega \ ; \\ \nabla^2 v(\mathbf{x}) &= 0 \ , & \mathbf{x} &\in \Omega \ , \\ v(\mathbf{x}) &= \beta(\mathbf{x}) \ , & \mathbf{x} &\in \partial\Omega \ . \end{aligned} \tag{6.2-4}$$

Thus a single fourth-order problem reduces to a coupled pair of second-order problems. The boundary-value problem for Equation (6.2-1) with clamped-edge conditions admits no such factorization. As we shall see, this observation has significant implications for discrete solution techniques.

Finite-difference approximation: single-equation approach.

We can derive finite-difference approximations to Equation (6.2-1) through straightforward application of the central difference operators defined in Section 2.6. Let us distinguish these operators acting in the x- and y-directions by denoting them as δ_x and δ_y. Assuming that the domain Ω is a rectangle $(a, b) \times (c, d)$, we can establish a two-dimensional grid $\Delta = \Delta_x \times \Delta_y$, where $\Delta_x : (a =) x_o < \cdots < x_n (= b)$ and $\Delta_y : (c =) y_o < \cdots < y_m (= d)$ as in Section 2.5. For convenience, let Δ be uniform in each coordinate direction, so that $x_i - x_{i-1} = h$ and $y_j - y_{j-1} = k$. Thus we

can approximate the operator $\nabla^4 = \partial^4/\partial x^4 + 2\partial^4/\partial x^2 \partial y^2 + \partial^4/\partial y^4$ by writing difference analogs for each term as follows:

$$\begin{aligned}\left.\frac{\partial^4 u}{\partial x^4}\right|_{(x_i,y_j)} &= \frac{1}{h^4}\delta_x^2(\delta_x^2 u_{i,j}) + O(h^2) \\ &= \frac{1}{h^4}\delta_x^2(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) + O(h^2) \\ &= \frac{1}{h^4}(u_{i-2,j} - 4u_{i-1,j} + 6u_{i,j} - 4u_{i+1,j} \\ &\quad + u_{i+2,j}) + O(h^2) \ .\end{aligned}$$

Similarly,

$$\begin{aligned}\left.\frac{\partial^4 u}{\partial y^4}\right|_{(x_i,y_j)} &= \frac{1}{k^4}(u_{i,j-2} - 4u_{i,j-1} + 6u_{i,j} \\ &\quad - 4u_{i,j+1} + u_{i,j+2}) + O(k^2) \ .\end{aligned}$$

Finally,

$$\begin{aligned}\left.\frac{\partial^4 u}{\partial x^2 \partial y^2}\right|_{(x_i,y_j)} &= \frac{1}{h^2}\delta_x^2\left(\frac{1}{k^2}\delta_y^2 u_{i,j}\right) + O(h^2 + k^2) \\ &= \frac{1}{h^2 k^2}\Big[u_{i+1,j+1} + u_{i-1,j+1} + u_{i-1,j-1} + u_{i+1,j-1} \\ &\quad - 2\left(u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1}\right) + 4u_{i,j}\Big] \\ &\quad + O(h^2 + k^2) \ .\end{aligned}$$

Combining these approximations, we arrive at a second-order difference analog for $\nabla^4 u = 0$, each equation of which couples 13 unknown nodal values of u. Figure 6-1 illustrates a typical difference molecule for the grid point (x_i, y_j), along with the weights corresponding to each nearby node in the difference approximation.

It is clear from this diagram that a typical row in the matrix arising from this discretization will contain 13 nonzero entries.

A difference molecule of this size leads to rather special considerations for treating boundary conditions, since boundary data will affect all difference equations centered no more than one node away from $\partial\Omega$. To accommodate the coupling over as many as five adjacent nodes, as shown in Figure 6-1, let us establish a layer of fictitious nodes around the boundary, as depicted for a representative boundary segment in Figure 6-2. We must now establish a one-to-one correspondence between finite-difference equations and nodal unknowns, using the boundary conditions. Consider first the simply-supported plate conditions (6.2-2). At a typical boundary

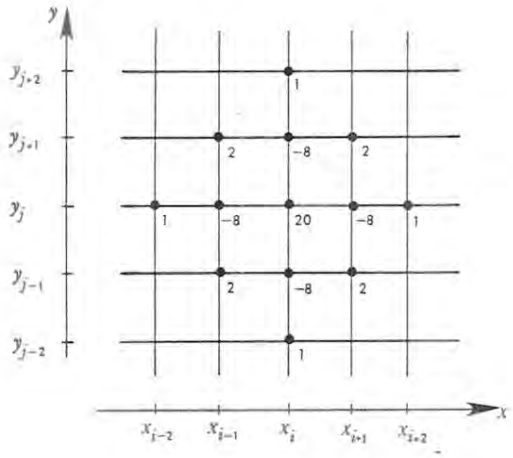

Figure 6-1. Thirteen-point difference molecule for the biharmonic operator ∇^4 in two dimensions, with numerical weights assigned to each node.

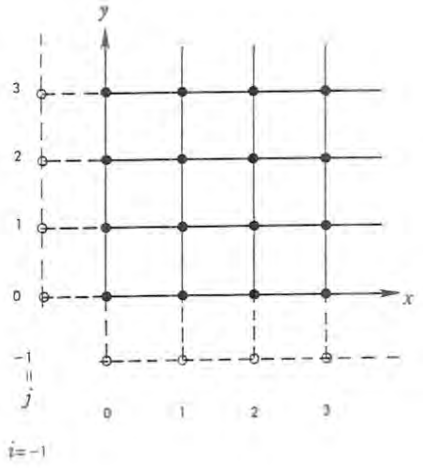

Figure 6-2. Arrangement of fictitious nodes near the boundary for the finite-difference analog of the biharmonic equation.

node like (x_0, y_0) or (x_0, y_2) the Dirichlet boundary condition $u(\mathbf{x}) = \alpha(\mathbf{x})$ gives explicit equations for the corresponding nodal value $u_{0,0}$ or $u_{0,2}$. The difficulty arises when we try to write the 13-point difference approximation along the interior layer of nodes, such as (x_1, y_1), (x_2, y_1), (x_1, y_2), and so forth, that lie adjacent to the boundary. Here we need to use the fictitious nodes to accommodate the fact that the difference equation calls for values of u at nodes lying outside the actual boundary $\partial\Omega$ of the domain. For example, when the grid meshes h and k in the x- and y-directions are equal, the difference equation associated with the unknown value $u_{1,2}$ at the node (x_1, y_2) requires

$$\begin{aligned} &20u_{1,2} - 8(u_{1,3} + u_{2,2} + u_{1,1} + u_{0,2}) \\ &+ 2(u_{0,3} + u_{2,3} + u_{2,1} + u_{0,1}) \\ &+ u_{1,4} + u_{3,2} + u_{1,0} + u_{-1,2} = 0 \ . \end{aligned} \tag{6.2-5}$$

Having added these fictitious nodes, we now need to ensure that they do not spoil the balance between equations and unknowns. Since there are difference equations centered at the nodal unknown $u_{1,3}$, $u_{2,2}$, $u_{1,1}$, $u_{2,3}$, $u_{2,1}$, $u_{1,4}$, and $u_{3,2}$ and Dirichlet boundary conditions assigning values to $u_{0,1}$, $u_{0,2}$, $u_{0,3}$, and $u_{0,1}$, we only need to eliminate the fictitious unknown $u_{-1,2}$ from Equation (6.2-5) to guarantee a one-to-one correspondence between equations and unknowns. Applying the difference approximation

$$\frac{1}{h^2}(u_{-1,2} + u_{0,3} + u_{1,2} + u_{0,1} - 4u_{0,2})$$

$$\simeq \nabla^2 u(x_0, y_2) = \beta(x_0, y_0)$$

to the second-order boundary condition, we obtain

$$u_{-1,2} = h^2\beta(x_0, y_0) + 4u_{0,2} - u_{0,3} - u_{1,2} - u_{0,1} \ .$$

Substituting this identity into Equation (6.2-5) yields an equation involving only interior nodal values and known boundary values. Keeping the unknowns on the left, this equation becomes

$$\begin{aligned} &19u_{1,2} - 8(u_{1,3} + u_{2,2} + u_{1,1}) + 2(u_{2,3} + u_{2,1}) + u_{1,4} + u_{3,2} \\ &= 4\alpha(x_0, y_2) - \alpha(x_0, y_3) - \alpha(x_0, y_1) - h^2\beta(x_0, y_0) \ . \end{aligned}$$

This same technique applies to the elimination of fictitious nodal values from equations centered at other nodes along the interior layer.

A similar strategy works in the case of clamped-edge conditions (6.2-3). To eliminate the fictitious variable $u_{-1,2}$ from Equation (6.2-4) in this

instance, we invoke a second-order difference approximation to the normal-derivative condition,

$$\frac{1}{2h}(u_{1,2} - u_{-1,2}) \simeq \left.\frac{\partial u}{\partial n}\right|_{(x_0,y_2)} = 0 .$$

This yields $u_{-1,2} = u_{1,2}$, so after accounting for the homogeneous Dirichlet boundary data we can rewrite Equation (6.2-5) as

$$21u_{1,2} - 8(u_{1,3} + u_{2,2} + u_{1,1}) + 2(u_{2,3} + u_{2,1}) + u_{1,4} + u_{3,2} = 0 ,$$

an equation involving no fictitious nodes.

The discretizations based on the 13-point difference molecule just described yield large, sparse sets of linear equations having the matrix form $\mathbf{Au} = \mathbf{b}$, where $\mathbf{A}$ is the coefficient matrix, $\mathbf{u}$ is the vector of unknown nodal values, and $\mathbf{b}$ is a vector of known boundary information. The construction of this matrix equation is no different in principle from analogous constructions presented earlier, for example, in Chapter Three. However, in practical computations the sensitivity of $\mathbf{u}$ to perturbations in the matrix entries or in the forcing vector $\mathbf{b}$ can be a crucial consideration. In Chapter Three we encountered the important notion that, when the condition number $\text{cond}(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$ is very large, small errors in the entries of $\mathbf{A}$ or $\mathbf{b}$ can produce large errors in the computed values of $\mathbf{u}$. This notion is the basis for an even more important observation regarding the numerical solution of the biharmonic equation.

Consider first, for simplicity's sake, the usual finite-difference solution of Poisson's equation $\nabla^2 u = f$ on a rectangle using a two-dimension-al grid of uniform mesh. As we saw in Chapter Three, this problem yields a sparse linear system $\mathbf{A}_1\mathbf{u} = \mathbf{b}_1$ in which the matrix $\mathbf{A}_1$ is symmetric. For such matrices, as we have seen in Section 2.9, the Euclidean norm is $\|\mathbf{A}_1\|_2 = |\lambda_{\max}|$, the magnitude of the eigenvalue of $\mathbf{A}_1$ lying furthest from the origin. Since the eigenvalues of $\mathbf{A}_1^{-1}$ are the reciprocals of those of $\mathbf{A}_1$, it follows that $\|\mathbf{A}_1^{-1}\|_2 = 1/|\lambda_{\min}|$, which is finite provided $\mathbf{A}_1$ is nonsingular. Thus the condition number for difference analogs to Poisson's equation is $\text{cond}(\mathbf{A}_1) = |\lambda_{\max}/\lambda_{\min}|$. One can show that, for Poisson problems on uniform grids, $\lambda_{\max} = \mathcal{O}(h^{-2})$ and $\lambda_{\min} = \mathcal{O}(1)$ as $h \to 0$. Thus $\text{cond}(\mathbf{A}_1) = \mathcal{O}(h^{-2})$ as $h \to 0$, and as a result finite-difference approximations on fine grids tend to yield poorly conditioned matrix equations. In practice, overcoming this difficulty requires preconditioning, iterative improvement, or high-precision machine arithmetic, all of which imply greater computational expense.

In the case of the biharmonic equation the probem is even worse. In this case the 13-point discretization of $\nabla^4 u = \nabla^2(\nabla^2 u)$ typically yields a symmetric matrix $\mathbf{A}$ whose eigenvalues are *squares* of the eigenvalues for

the matrix $\mathbf{A}_1$ of Poisson's problem on the same grid (Birkhoff and Lynch, 1984, Section 3.8). Thus for the biharmonic equation we can expect

$$\text{cond}(\mathbf{A}) = \frac{|\lambda_{\max}|}{|\lambda_{\min}|} = \frac{\mathcal{O}(h^{-4})}{\mathcal{O}(1)} = \mathcal{O}(h^{-4}).$$

This implies an even faster increase in condition number for fine grids than with second-order problems and a concomitantly faster increase in computational expense.

Finite-difference approximation: coupled-equation approach.

The reasoning just given suggests that it may be more efficient to solve the factored form of the biharmonic equation whenever this is possible. The strategy here is to replace a single set of nodal equations having condition number $\mathcal{O}(h^{-4})$ by a possibly sparser set of roughly twice as many equations having a much smaller condition number $\mathcal{O}(h^{-2})$. To do this, let us return to Equations (6.2-4), which we rewrite as follows:

$$\begin{aligned} \nabla^2 \begin{bmatrix} u \\ v \end{bmatrix} &= \begin{bmatrix} v \\ 0 \end{bmatrix} \quad \text{on} \quad \Omega, \\ \begin{bmatrix} u \\ v \end{bmatrix} &= \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad \text{on} \quad \partial\Omega\,. \end{aligned}$$

Now we can simply approximate $\nabla^2 u$ and $\nabla^2 v$ using the standard second-order difference analogs on the same grid Δ defined for the unfactored form treated above. Thus

$$\begin{aligned} \nabla^2 u = &\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} \\ &+ \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{k^2} + \mathcal{O}(h^2 + k^2), \end{aligned}$$

and similarly for v. The resulting difference equations furnish a collection of algebraic equations having the form

$$\begin{aligned} \frac{1}{h^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) + \frac{1}{k^2}(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}) &= v_{i,j}, \\ \frac{1}{h^2}(v_{i-1,j} - 2v_{i,j} + v_{i+1,j}) + \frac{1}{k^2}(v_{i,j-1} - 2v_{i,j} + v_{i,j+1}) &= 0, \end{aligned}$$

at each interior node (x_i, y_j). While this approach requires the solution of twice as many equations as the 13-point difference formulations, the

advantages in matrix sparseness and slower degradation in matrix condition number as $h \to 0$ often outweigh the disadvantages associated with increases in matrix order.

Finite-element approximations: single-equation approach.

With finite-element aproximations we face a similar choice between solving Equation (6.2-1) as one fourth-order PDE or as a coupled system of two second-order PDEs. Let us begin by examining the first option, using both the simply supported plate conditions (6.2-2) and the clamped-plate boundary conditions (6.2-3).

Before selecting a particular basis for the trial function $\hat{u}(\mathbf{x})$, let us examine the Galerkin formalism to see what requirements $\hat{u}$ will have to satisfy. Assume $\hat{u}$ has an expansion of the form

$$\hat{u}(\mathbf{x}) = u_\partial(\mathbf{x}) + \sum_{i=1}^{M} u_i \phi_i(\mathbf{x}). \tag{6.2-6}$$

Here, the basis functions $\phi_i(\mathbf{x})$, $i = 1, \ldots, M$, satisfy homogeneous boundary conditions, that is, $\phi_i(\mathbf{x}) = 0$ for $\mathbf{x} \in \partial\Omega$. We shall see in a moment what boundary values to impose on the function $u_\partial(\mathbf{x})$. Given such a trial function, the Galerkin integral equations are as follows:

$$\int_\Omega \nabla^4 \hat{u}(\mathbf{x}) \phi_j(\mathbf{x})\, d\mathbf{x} = 0, \quad j = 1, \ldots, M.$$

To reduce the formal smoothness constraints on $\hat{u}$ implied in these equations, we apply Green's theorem to shift differentiation from the trial function to the weighting functions $\phi_j(\mathbf{x})$. One application of Green's theorem yields

$$-\int_\Omega \nabla[\nabla^2 \hat{u}(\mathbf{x})] \cdot \nabla \phi_j(\mathbf{x})\, d\mathbf{x} + \oint_{\partial\Omega} \phi_j(\mathbf{x}) \nabla[\nabla^2 \hat{u}(\mathbf{x})] \cdot \mathbf{n}\, d\mathbf{x} = 0,$$

and another gives

$$\int_\Omega \nabla^2 \hat{u}(\mathbf{x})\, \nabla^2 \phi_j(\mathbf{x})\, d\mathbf{x} + \oint_{\partial\Omega} \phi_j(\mathbf{x}) \nabla[\nabla^2 \hat{u}(\mathbf{x})] \cdot \mathbf{n}\, d\mathbf{x} - \oint_{\partial\Omega} \nabla^2 \hat{u}(\mathbf{x}) \nabla \phi_j(\mathbf{x}) \cdot \mathbf{n}\, d\mathbf{x} = 0. \tag{6.2-7}$$

Since the trial function $\hat{u}$ and the weighting functions ϕ_j possess the same degree of smoothness, there is no point in shifting differentiation between the two any further.

Before examining the treatment of boundary conditions, let us digress for a moment to make an observation that has important implications for coding. Equation (6.2-7) implies that the second derivatives $\nabla^2\phi_j(\mathbf{x})$ and $\nabla^2\hat{u}(\mathbf{x})$ must have, at worst, jump discontinuities between finite elements Ω_e if the first integral is to admit decomposition into a sum of integrals over individual elements, as in

$$\int_\Omega \nabla^2\hat{u}\,\nabla^2\phi_j\,d\mathbf{x} = \sum_e \int_{\Omega_e} \nabla^2\hat{u}\,\nabla^2\phi_j\,d\mathbf{x}.$$

Such a decomposition is highly desirable, since it allows elementwise computation of the matrix entries in a computer code. Therefore, to guarantee the validity of this decomposition, we must choose basis functions $\{\phi_j\}_{j=1}^N$ that have continuous gradients, that is, that belong to $C^1(\Omega)$. In general, basis functions that allow elementwise decomposition of the Galerkin volume integrals are called **conforming elements**.

Now we return to the issue of boundary conditions. The presence of the boundary integrals in the Galerkin equations (6.2-7) implies that any boundary conditions having the forms

$$\begin{aligned} \nabla^2 u(\mathbf{x}) &= \beta(\mathbf{x}) \quad \text{on} \quad \partial\Omega, \\ \nabla[\nabla^2 u(\mathbf{x})]\cdot\mathbf{n} &= \gamma(\mathbf{x}) \quad \text{on} \quad \partial\Omega, \end{aligned}$$

can be accommodated via straightforward substitution of the functions β and γ in the boundary terms. Therefore these are natural boundary conditions. However, no such mechanism exists for imposing the lower-order boundary conditions

$$\begin{aligned} u(\mathbf{x}) &= \alpha(\mathbf{x}) \quad \text{on} \quad \partial\Omega, \\ \nabla u(\mathbf{x})\cdot\mathbf{n} &= \xi(\mathbf{x}) \quad \text{on} \quad \partial\Omega, \end{aligned}$$

and our only recourse is to impose them a priori in the construction of the trial function $\hat{u}$. Therefore, these are essential boundary conditions.

In view of these observations, the Hermite bicubic interpolating functions constitute a feasible choice of basis functions for Galerkin approximations to the biharmonic equation. With this choice the trial function will look like

$$\begin{aligned} \hat{u}(\mathbf{x}) = u_\partial(\mathbf{x}) + \sum_{i=1}^N [&u_i\phi_{00i}(\mathbf{x}) + u_i^{(x)}\phi_{10i}(\mathbf{x}) \\ &+ u_i^{(y)}\phi_{01i}(\mathbf{x}) + u_i^{(xy)}\phi_{11i}(\mathbf{x})]. \end{aligned}$$

In this representation the functions $\phi_{jki}(\mathbf{x})$ are tensor products of the usual one-dimensional Hermite cubics described in Section 2.4; specifically,

$\phi_{jki}(x,y) = h_i^j(x)h_i^k(y)$. The unknown coefficients $u_i, u_i^{(x)}, u_i^{(y)}$, and $u_i^{(xy)}$ stand for the values of $\hat{u}, \partial\hat{u}/\partial x, \partial\hat{u}/\partial y$, and $\partial^2\hat{u}/\partial x\partial y$, respectively, at the node $\mathbf{x}_i$. Recall from Chapter Two that the interpolation error associated with this type of trial function is $\mathcal{O}(h^4 + k^4)$.

For example, for simply supported plate conditions, we accommodate essential boundary conditions in the trial function of Equation (6.2-6) by defining the boundary function

$$u_\partial(\mathbf{x}) = \sum_\partial u_i\, \phi_{00i}(\mathbf{x}).$$

Here, the notation $\sum_\partial$ indicates the sum over boundary nodes $\mathbf{x}_i$. The coefficients u_i in this sum are known boundary values given by $u_i = \alpha(\mathbf{x}_i)$. The remaining condition $\nabla^2 u = \beta(\mathbf{x})$ on $\partial\Omega$ is a natural boundary condition. Thus we need not explicitly incorporate it into the definition of $\hat{u}$, since we can use it to compute boundary integrals.

For clamped-edge conditions the construction of $u_\partial(\mathbf{x})$ involves only slightly more complication, depending on the shape of the domain Ω. If Ω is a rectangle $(a,b)\times(c,d)$ as above, then $\partial\Omega$ consists of line segments parallel to the x- and y-axes. If $\sum_{\partial_x}$ and $\sum_{\partial_y}$ stand for sums over boundary nodes along the segments parallel to the x- and y- axes, respectively, then

$$\begin{aligned} u_\partial(\mathbf{x}) = &\sum_\partial u_i\phi_{00i}(\mathbf{x}) + \sum_{\partial_x} u_i^{(y)}\phi_{01i}(\mathbf{x}) \\ &+ \sum_{\partial_y} u_i^{(x)}\phi_{10i}(\mathbf{x}) + \sum_{\text{corners}} u_i^{(xy)}\phi_{11i}(\mathbf{x}). \end{aligned}$$

Since the clamped-edge conditions force $u(\mathbf{x}) = \nabla u(\mathbf{x})\cdot\mathbf{n} = 0$ along $\partial\Omega$, each of the boundary nodal values u_i and the nodal normal derivatives $u_i^{(y)}, u_i^{(x)}$ appearing in these sums vanishes. Moreover, differentiating $\nabla u(\mathbf{x})\cdot\mathbf{n} = 0$ tangentially along $\partial\Omega$ shows that $\partial^2 u/\partial x\partial y = 0$ on $\partial\Omega$, so that the nodal cross-derivatives $u_i^{(xy)}$ also vanish on the boundaries. Thus, in the clamped-edge case, $u_\partial(\mathbf{x}) = 0$, and the remainder of the trial function $\hat{u}$ involves only the interior nodal values u_i and the unknown tangential derivatives $u_i^{(x)}$ or $u_i^{(y)}$ along $\partial\Omega$.

It is worth mentioning that other continuously differentiable finite-element bases for the biharmonic equation have appeared in the literature. In particular there are at least three approaches using triangular elements (Strang and Fix, 1973, Section 1.9). The first and most straightforward of these is to use continuously differentiable piecewise quintic polynomials defined on triangles. Such quintics have 21 degrees of freedom. Of these degrees of freedom, 18 must determine the values of $\hat{u}, \partial\hat{u}/\partial x, \partial\hat{u}/\partial y, \partial^2\hat{u}/\partial x^2, \partial^2\hat{u}/\partial x\partial y$, and $\partial^2\hat{u}/\partial y^2$ at the vertices of each

triangle to force continuous differentiability at the vertices. The remaining three degrees of freedom determine the mid-side values of the normal derivative $\partial\hat{u}/\partial n$ to guarantee continuous differentiability across the edges of the triangle. Figure 6-3a illustrates this element.

For the full quintic element the interpolation error is $\mathcal{O}(h^6)$, where, as above, h signifies the maximum dimension of a triangle in the grid.

The second approach using triangles is to eliminate the three off-vertex degrees of freedom in the quintic triangle by forcing the quintic to reduce to a cubic along the edges. These reduced quintics involve fewer unknowns per element than the standard quintic, 18 instead of 21, as shown in Figure 6-3b. Reduced quintics therefore required somewhat less computational effort for a given triangulation than full quintics. The penalty paid for this gain is a slight decrease in accuracy: The reduced quintic element has an interpolation error that is $\mathcal{O}(h^5)$.

The third continuously differentiable triangular element that we shall mention is the **Clough-Tocher element** (Clough and Tocher, 1966). The idea here is to divide each triangle into three "daughter" triangles, as shown in Figure 6-3c We then impose internal continuity constraints leaving only 12 values, defining $\hat{u}, \partial\hat{u}/\partial x, \partial\hat{u}/\partial y$ at the vertices and $\partial\hat{u}/\partial n$ at the mid-side nodes, as elemental degrees of freedom. To accomplish this, we allow each daughter triangle to be cubic in x and y, thus allowing for 10 initial degrees of freedom per daughter triangle and therefore 30 initial degrees of freedom for the "parent" triangle. By imposing common values of $\hat{u}, \partial\hat{u}/\partial x$, and $\partial\hat{u}/\partial y$ at each parent vertex and at the vertex common to the three daughter triangles, we arrive at constraints sufficient to eliminate 16 of the 30 degrees of freedom. Then, by requiring common values of $\partial\hat{u}/\partial n$ along the internal edges of the daughter triangles, we eliminate three additional degrees of freedom, leaving 12 degrees of freedom undetermined. For details of the algebra involved in this elimination we refer the reader to the original paper by Clough and Tocher (1966).

There is one final class of approaches worth mentioning in the single-equation formulation of Equation (6.2-1). The motivation for this approach is the desire to abandon the smoothness constraints on $\hat{u}(\mathbf{x})$ implied by equation (6.2-6). As we have seen, these constraints lead to elements requiring many degrees of freedom and thus to Galerkin matrices having undesirably large bandwidths. Finite-element basis funcitons that violate these smoothness constraints are called **nonconforming** elements. One example of a nonconforming element used in connection with the biharmonic equation is the **Morley element**, which is a quadratic polynomial defined over a triangular region. Such a polynomial has six degrees of freedom, which define the three nodal values at the vertices and the three normal derivatives at mid-side nodes. This smaller number of unknowns per element allows great computational efficiency compared to conforming

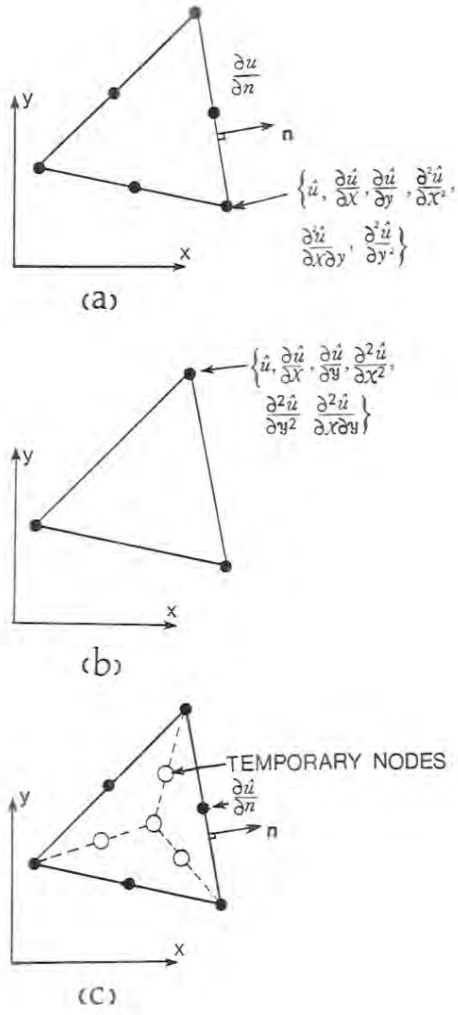

Figure 6-3. Continuously differentiable triangular finite elements: (a) the complete C^1 quintic element, (b) the reduced C^1 quintic element, (c) the Clough-Tocher element (after Lapidus and Pinder, 1982, p. 460.)

elements. However, far from being continuously differentiable, trial functions using bases of Morley elements are not even continuous across the edges of triangles. Therefore, elementwise calculation of the Galerkin integrals is no longer formally valid. Nevertheless, Galerkin approximations using this element still yield convergent approximations to Equation (6.2-1). The analysis of nonconforming finite-element methods lies beyond the scope of this book, and we refer the reader to Griffiths and Mitchell (1984) for a review of the subject.

Finite-element approximations: coupled-equation approach.

We close our discussion of the biharmonic equation with an introduction to finite-element approximations for the coupled equations (6.2-4). In the interest of slightly more generality, let us consider a nonhomogeneous version of the biharmonic equation, $\nabla^4 u = f$. In this case the coupled second-order PDEs become

$$\nabla^2 \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} v \\ f \end{bmatrix} \quad \text{on} \quad \Omega.$$

Thus we seek two trial functions $\hat{u}(\mathbf{x})$ and $\hat{v}(\mathbf{x})$. It will turn out that, compared with the single-equation approach, this two-equation formulation relaxes the smoothness constraints on the finite-element bases for $\hat{u}$ and $\hat{v}$.

To see how, let us consider trial functions of the form

$$\hat{u}(\mathbf{x}) = u_\partial(\mathbf{x}) + \sum_{i=1}^{N} u_i \phi_i(\mathbf{x}),$$

$$\hat{v}(\mathbf{x}) = v_\partial(\mathbf{x}) + \sum_{j=1}^{M} v_j \psi_j(\mathbf{x}).$$

Notice that we have allowed for different bases $\{\phi_i\}_{i=1}^N$ and $\{\psi_j\}_{j=1}^M$ in the construction of $\hat{u}$ and $\hat{v}$. Such finite-element formulations of factored PDE's using separate expansions of the unknown u and one of its derivatives (in this case $\nabla^2 u$) are called **mixed methods**. As with Galerkin formulations for single equations, each of the interior basis functions $\phi_i(\mathbf{x}), \psi_j(\mathbf{x})$ must satisfy homogeneous boundary conditions. This means $\phi_i(\mathbf{x}) = 0$ and $\psi_j(\mathbf{x}) = 0$, whenever $\mathbf{x} \in \partial\Omega$. The Galerkin integral equations for these trial functions become

$$\int_\Omega (\nabla^2 \hat{u} - \hat{v}) \psi_k \, d\mathbf{x} = 0, \quad k = 1, \ldots, M,$$

$$\int_\Omega (\nabla^2 \hat{v} - f)\phi_\ell \, d\mathbf{x} = 0, \quad \ell = 1, \ldots, N.$$

(The particular arrangement of weighting functions in these equations arises from the variational formulation of the problem; for details we refer the reader to Carey and Oden, 1983, Section 3.3.) Making use of Green's theorem, we get

(6.2-8a)
$$\begin{aligned} &- \int_\Omega \nabla \hat{u} \cdot \nabla \psi_k \, d\mathbf{x} - \int_\Omega \hat{v} \psi_k \, d\mathbf{x} \\ &+ \oint_{\partial\Omega} \psi_k \nabla \hat{u} \cdot \mathbf{n} \, d\mathbf{x} = 0, \quad k = 1, \ldots, M, \\ &- \int_\Omega \nabla \hat{v} \cdot \nabla \phi_\ell \, d\mathbf{x} - \int_\Omega f \phi_\ell \, d\mathbf{x} \\ &+ \oint_{\partial\Omega} \phi_\ell \nabla \hat{v} \cdot \mathbf{n} \, d\mathbf{x} = 0, \quad \ell = 1, \ldots, N. \end{aligned}$$

Now we can see how the mixed formulation allows us to use trial functions satisfying less stringent smoothness constraints. The integrals over Ω in Equations (6.2-8a) will admit elementwise decompositions of the form

$$\int_\Omega (\cdot) d\mathbf{x} = \sum_e \int_{\Omega_e} (\cdot) d\mathbf{x}$$

provided the gradients of the basis functions are at worst jump-discontinuous at element boundaries. Therefore the trial functions $\hat{u}$ and $\hat{v}$ need only belong to $C^0(\Omega)$ for the finite-element method to be conforming. In particular, we can choose tensor-product Lagrange bases or piecewise-planar triangles to form our trial functions. Thus the mixed formulation enjoys less restrictive smoothness requirements than the fourth-order formulation.

Equations (6.2-8a) also show that boundary conditions specifying values of $\partial u/\partial n = \nabla u \cdot \mathbf{n}$ and $\partial v/\partial n = \nabla(\nabla^2 u) \cdot \mathbf{n}$ along $\partial\Omega$ will be natural boundary conditions for the coupled-equation approach. On the other hand, boundary conditions specifying u and $v = \nabla^2 u$ on $\partial\Omega$ will be essential boundary conditions, which we must incorporate into the definitions of $\hat{u}(\mathbf{x})$ and $\hat{v}(\mathbf{x})$ through the boundary terms $u_\partial(\mathbf{x})$ and $v_\partial(\mathbf{x})$.

Equations (6.2-8a) give rise to matrix equations having an interesting structure. Assume for simplicity that the boundary conditions are homogeneous, so that

$$u = 0 \quad \text{on} \quad \partial\Omega,$$

$$\nabla^2 u = v = 0 \quad \text{on} \quad \partial\Omega,$$

and hence $u_\partial(\mathbf{x}) \equiv v_\partial(\mathbf{x}) \equiv 0$ in the trial functions. The Galerkin equations (6.2-7) then expand to give

$$\text{(6.2-8b)} \qquad \sum_{i=1}^{N} u_i \int_\Omega \nabla\phi_i \cdot \nabla\psi_k \, d\mathbf{x} + \sum_{j=1}^{M} v_j \int_\Omega \psi_j \psi_k \, d\mathbf{x}$$

$$= \oint_{\partial\Omega} \psi_k \nabla\hat{u} \cdot \mathbf{n} \, d\mathbf{x}, \quad k = 1, \ldots, N \, ,$$

$$\text{(6.2-8c)} \qquad \sum_{j=1}^{M} v_j \int_\Omega \nabla\psi_j \cdot \nabla\phi_\ell \, d\mathbf{x} = -\int_\Omega f\phi_\ell \, d\mathbf{x}$$

$$+ \oint_{\partial\Omega} \phi_\ell \nabla\hat{v} \cdot \mathbf{n} \, d\mathbf{x}, \quad \ell = 1, \ldots, M \, .$$

Notice that the boundary integrals in each of these sets of equations all vanish, since the interior basis functions $\phi_\ell(\mathbf{x})$ and $\phi_k(\mathbf{x})$ vanish on $\partial\Omega$. Now denote by $\mathbf{K}$ the $M \times N$ matrix whose (i,k)-th entry is $\int_\Omega \nabla\phi_i \cdot \nabla\psi_k \, d\mathbf{x}$, by $\mathbf{M}$ the $M \times M$ matrix whose (j,k)-th entry is $\int_\Omega \psi_j\psi_k \, d\mathbf{x}$, by $\mathbf{u}$ and $\mathbf{v}$ the column vectors containing the nodal values $u_1, \ldots, u_N$ and $v_1, \ldots, v_M$, respectively, and by $\mathbf{f}$ the M-dimensional vector whose ℓ-th entry is $\int_\Omega f\phi_\ell \, d\mathbf{x}$. Then Equations (6.2-8) assume the form

$$\begin{bmatrix} \mathbf{K} & \mathbf{M} \\ \mathbf{0} & \mathbf{K}^\top \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{f} \end{bmatrix} .$$

This matrix structure has the advantage that it is already "almost" upper triangular. This property, together with the fact that the matrices $\mathbf{K}$ and $\mathbf{M}$ are typically sparse, makes it possible to envision highly efficient algorithms for solving the matrix equations arising from mixed methods. These observations serve to mitigate further the disadvantages associated with solving for two unknowns instead of one at every grid point.

6.3. Nonlinear Problems.

Many of the PDEs arising in practical problems are nonlinear. Typically, the nonlinearity owes its existence to dependencies of material properties or forcing functions on the unknown in the problem. This section reviews several approaches to the numerical solution of nonlinear PDEs, relying on the physically motivated examples described below. Several exercises at the end of this chapter introduce still other approaches to the discretization of nonlinear PDEs.

Steady, nonlinear heat flow.

To begin with, let us examine two types of nonlinearity that commonly occur in steady-state problems. We start with the energy balance for heat flow developed in Sections 1.4 and 4.1. When heat sources are present, Equation (4.1-6) becomes

$$\rho c_v \frac{\partial T}{\partial t} = \nabla \cdot (k_H \nabla T) + \rho h, \tag{6.3-1}$$

where T is the unknown temperature, ρ is the mass density, c_v is the heat capacity of the material, and h represents the external supply of heat. In a steady state, $\partial T/\partial t = 0$, and we can write Equation (6.3-1) as

$$\nabla \cdot (k_H \nabla T) = -\rho h. \tag{6.3-2}$$

In simple cases when k_H is constant, the heat supply term may still depend on the temperature T. Such a dependence occurs, for example, in systems governed by thermostats or in materials whose density ρ exhibits significant dependence on temperature. Under these assumptions, Equation (6.3-2) becomes

$$\nabla^2 T = -\rho(T)h(T)/k_H, \tag{6.3-3}$$

which is a nonlinear version of Poisson's equation.

If, in addition, the heat flux coefficient k_H depends on T, then the terms involving spatial derivatives in Equation (6.3-2) also become nonlinear, and we are left with the elliptic equation

$$\nabla \cdot [k_H(T)\nabla T] = -\rho(T)h(T). \tag{6.3-4}$$

Thus in steady-state heat-flow problems nonlinearity can occur either in the forcing term modeling heat supplies or through the material property governing the rate of heat flux.

Nonlinear diffusion.

Next, consider a transient problem involving the diffusion of a solute in a fluid, as examined in Section 1.5, when we relax the assumption that the solute transport is passive. In the absence of advection and chemical reactions, the species mass balance equation for the dissolved solute is

$$\frac{\partial}{\partial t}(\rho \omega^S) + \nabla \cdot \mathbf{j}^S = 0. \tag{6.3-5}$$

Let us assume, as in Section 1.5, that the diffusive flux $\mathbf{j}^S$ obeys Fick's law, only now we shall allow the diffusion coefficient K^S to depend on the unknown mass fraction ω^S. Thus, we shall assume

$$\mathbf{j}^S = -K^S(\omega^S)\nabla(\rho\omega^S); \quad K^S > 0.$$

Also, since we are abandoning the assumption that the solute transport is passive, we may as well allow the overall mixture density ρ to vary with the amount of solute present by letting $\rho = \rho(\omega^S)$. However, we shall assume that gradients in density arising from this dependence are small compared with gradients in ω^S. Using these constitutive assumptions, we find that Equation (6.3-5) reduces to the nonlinear parabolic equation

$$\frac{\partial}{\partial t}\left[\rho(\omega^S)\omega^S\right] - \nabla \cdot \left[\rho(\omega^S)K^S(\omega^S)\,\nabla\omega^S\right] = 0.$$

If we now recognize that the combination $\rho(\omega^S)\omega^S$ is just the solute density $\rho^S(\omega^S)$ and call $\rho(\omega^S)K^S(\omega^S) = k_D(\omega^S)$, then we arrive at the slightly simpler form,

$$\frac{\partial}{\partial t}\left[\rho^S(\omega^S)\right] - \nabla \cdot \left[k_D(\omega^S)\,\nabla\omega^S\right] = 0. \tag{6.3-6}$$

This equation exhibits nonlinearity both in the accumulation term and in the flux term.

The remainder of this section is devoted to the discussion of methods for solving each of the types of problems just cited. There are three such problems: the **nonlinear Poisson equation** generalized from Equation (6.3-3),

$$\nabla^2 u = f(u); \tag{6.3-7}$$

the **nonlinear steady heat flow equation** generalized from Equation (6.3-4),

$$\nabla \cdot [K(u)\,\nabla u] = f(u); \tag{6.3-8}$$

and the **nonlinear diffusion equation** generalized from Equation (6.3-6),

$$\frac{\partial}{\partial t}[c(u)] - \nabla \cdot [K(u)\,\nabla u] = 0. \tag{6.3-9}$$

While these equations are representative of many nonlinear forms that occur in science and engineering, they by no means exhaust all of the possibilities. Notice in particular that Equations (6.3-7) through (6.3-9) do

not include nonlinear equations applicable to nondissipative systems. In this regard, we have already discussed nonlinear hyperbolic PDEs, such as Burgers' equation, in Chapter Five. In Section 6.5 we shall see how considerations similar to those discussed in Chapter Five arise in a nonlinear coupled system governing oil-reservoir flows.

The nonlinear Poisson equation.

The nonlinear Poisson problem is in many ways the simplest of the four examples we shall discuss, so let us attack it first. Consider, for example, a Galerkin finite-element approximation to the homogeneous boundary-value problem

$$\nabla^2 u = f(u) \quad \text{in } \Omega, \tag{6.3-10}$$

$$u \equiv 0 \quad \text{on } \Omega.$$

If we replace the unknown function $u(\mathbf{x})$ by a trial function

$$\hat{u}(\mathbf{x}) = \sum_{i=1}^{N} u_i \, \phi_i(\mathbf{x}), \tag{6.3-11}$$

then applying the method of weighted residuals with the basis functions $\phi_1, \ldots, \phi_N$ serving as weighting functions yields the Galerkin integral equations

$$\int_\Omega \left[\nabla^2 \hat{u} - f(\hat{u})\right] \phi_j \, d\mathbf{x} = 0, \quad j = 1, \ldots, N. \tag{6.3-12}$$

Using Green's theorem and observing that the boundary contributions vanish since $\hat{u} = 0$ on $\partial\Omega$, we obtain

$$\sum_{i=1}^{N} u_i \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \, d\mathbf{x} + \int_\Omega f\left(\sum_{i=1}^{N} u_i \phi_i\right) \phi_j \, d\mathbf{x} = 0, \tag{6.3-13}$$

$$j = 1, \ldots, N.$$

These Galerkin equations have an equivalent matrix form

$$\mathbf{A}\mathbf{u} = -\mathbf{f}(\mathbf{u}), \tag{6.3-14}$$

where the (j,i)-th entry of the $N \times N$ matrix $\mathbf{A}$ is $\int_\Omega \nabla\phi_i \cdot \nabla\phi_j d\mathbf{x}$, the j-th entry of the vector $\mathbf{f}$ is $\int_\Omega f(\hat{u})\phi_j d\mathbf{x}$, and the i-th entry of $\mathbf{u}$ is the unknown coefficient u_i. Because the right side of Equation (6.3-14) depends on the unknown vector $\mathbf{u}$, we cannot solve this matrix equation by a single pass

through any of the linear matrix solution algorithms described in Sections 3.11 through 3.15. However, we can make use of these computationally attractive linear methods by adopting an *iterative* strategy in which each iteration requires the solution of a linear approximation to Equation (6.3-14).

As a simple example, suppose we begin with an initial guess $\mathbf{u}^{(0)}$ for the solution vector $\mathbf{u}$. We can use this guess to evaluate the forcing vector $\mathbf{f}$, giving a tentative value $\mathbf{f}(\mathbf{u}^{(0)})$ in the right side of Equation (6.3-14). Given this right side, we can solve the equation $\mathbf{A}\mathbf{u}^{(1)} = -\mathbf{f}(\mathbf{u}^{(0)})$ for a new iterative value $\mathbf{u}^{(1)}$ that, we hope, gives a better approximation to the true solution $\mathbf{u}$. We can proceed in this fashion, at each iteration using the most recent known iterative value $\mathbf{u}^{(m)}$ to solve for a new value $\mathbf{u}^{(m+1)}$ via the linear matrix equation

$$\mathbf{A}\mathbf{u}^{(m+1)} = -\mathbf{f}(\mathbf{u}^{(m)}). \tag{6.3-15}$$

This iterative scheme is an example of the method of **successive substitution.**

In practice, we can never expect $\mathbf{u}^{(m+1)}$ to equal the exact finite-element solution $\mathbf{u}$. But, assuming that the scheme produces sufficiently better approximations $\mathbf{u}^{(m+1)}$ at each iteration, we can run the iterations a finite number of times until the approximate solution satisfies some **convergence criterion.** For example, we might compute the **residual** $\mathbf{R}^{(m+1)} = \mathbf{A}\mathbf{u}^{(m+1)} + \mathbf{f}(\mathbf{u}^{(m+1)})$ at each iteration, stopping the iterations as soon as $\|\mathbf{R}^{(m+1)}\|$ falls below a prescribed tolerance in some norm. As an alternative, if we can estimate the **error** $\mathbf{e}^{(m+1)} = \mathbf{u}^{(m+1)} - \mathbf{u}$, we can stop iterating as soon as $\|\mathbf{e}^{(m+1)}\|$ is small enough.

Two questions immediately arise. First, when can we expect the iterative scheme to converge? That is, for which problems can we expect Equation (6.3-15) to yield an iterative sequence $\{\mathbf{u}^{(m)}\}_{m=0}^{\infty}$ such that $\mathbf{u}^{(m)} \to \mathbf{u}$ as $m \to \infty$? Second, if the iterative scheme does converge, at what rate does the error decrease? For many nonlinear problems in practice, these questions are quite difficult to answer with any precision. We can, however, outline a framework for the analysis of particular problems.

Consider first the question whether a scheme converges. Let us begin by writing Equation (6.3-15) in the equivalent form $\mathbf{u}^{(m+1)} = -\mathbf{A}^{-1}\mathbf{f}(\mathbf{u}^{(m)})$. This equation is a special case of the more general iterative scheme

$$\mathbf{u}^{(m+1)} = \mathbf{g}(\mathbf{u}^{(m)}), \tag{6.3-16}$$

where $\mathbf{g}$ is some function mapping N-vectors to N-vectors. By definition, such a function satisfies a **Lipschitz condition** of order L in a region $\mathcal{U}$ of N-space if there exists a positive constant L such that, whenever the vectors $\mathbf{v}$ and $\mathbf{w}$ lie in $\mathcal{U}$,

$$\|\mathbf{g}(\mathbf{v}) - \mathbf{g}(\mathbf{w})\| \leq L\|\mathbf{v} - \mathbf{w}\|,$$

in some norm. When the Lipschitz constant $L < 1$, the function $\mathbf{g}$ is called a **strict contraction**. To see how these concepts relate to the issue of convergence, observe that, according to Equation (6.3-14), the true solution to the finite-element discretization of the Poisson problem satisfies $\mathbf{u} = \mathbf{g}(\mathbf{u})$. In other words, $\mathbf{u}$ is a **fixed point** of the function $\mathbf{g}$. The following theorem, proved in Ortega and Rheinboldt (1970, p. 120), establishes the crucial connection between strict contractions and convergence of their associated iterative schemes under successive substitution:

Contraction Mapping Theorem. *Given a function* $\mathbf{g}$ *that maps* N*-vectors from a region* $\mathcal{U}$ *of* N*-space into* $\mathcal{U}$*, if* $\mathbf{g}$ *is a strict contraction on* $\mathcal{U}$*, then the iterative scheme (6.3-16) converges to a unique fixed point* $\mathbf{u} \in \mathcal{U}$ *for any initial guess* $\mathbf{u}^{(0)} \in \mathcal{U}$.

Therefore, to guarantee that the sucessive substitution scheme (6.3-16) converges, it suffices to find a region $\mathcal{U}$ of N-dimensional Euclidean space on which the function $\mathbf{g}$ is a strict contraction. For highly complex problems, this may be quite difficult to do rigorously, and it may be necessary to rely on experimental calculations to estimate where $\mathbf{g}$ will be a strict contraction.

In problems where we can estimate the Lipschitz constant L, we can also estimate the error at each iteration. Ortega and Rheinboldt (1970, p. 385) prove that, given a strict contraction as in the theorem just stated, the error $\mathbf{e}^{(m)} = \mathbf{u}^{(m)} - \mathbf{u}$ at the m-th iteration obeys the bound

$$\|\mathbf{e}^{(m)}\| \leq \frac{L}{L-1}\|\mathbf{u}^{(m)} - \mathbf{u}^{(m-1)}\|.$$

The right side of this inequality is computable at any iteration $m \geq 1$, so all we need to get error bounds in a computer code is an estimate of L. Whenever $\mathbf{g}$ is a differentiable function, it is possible to show (using the mean value theorem) that L is related to the Jacobian matrix $\mathbf{g}'(\mathbf{u})$ of $\mathbf{g}$ by the identity

$$L = \sup_{\mathbf{v} \in \mathcal{U}} \|\mathbf{g}'(\mathbf{v})\|,$$

that is, L is the least upper bound of $\|\mathbf{g}'(\mathbf{v})\|$ taken over all $\mathbf{v} \in \mathcal{U}$.

We can also estimate how fast the iterative scheme converges when $\mathbf{g}$ is differentiable. Since the true finite-element solution $\mathbf{u}$ is a fixed point of $\mathbf{g}$,

$$\begin{aligned}\mathbf{e}^{(m+1)} &= \mathbf{u}^{(m+1)} - \mathbf{u} = \mathbf{g}(\mathbf{u}^{(m)}) - \mathbf{g}(\mathbf{u}) \\ &= \mathbf{g}(\mathbf{u} + \mathbf{e}^{(m)}) - \mathbf{g}(\mathbf{u}).\end{aligned}$$

Therefore, in any norm,

$$\|\mathbf{e}^{(m+1)}\| = \|\mathbf{e}^{(m)}\| \frac{\|\mathbf{g}(\mathbf{u} + \mathbf{e}^{(m)}) - \mathbf{g}(\mathbf{u})\|}{\|\mathbf{e}^{(m)}\|},$$

and we find that the ratio of successive error norms obeys

$$\frac{\|\mathbf{e}^{(m+1)}\|}{\|\mathbf{e}^{(m)}\|} = \frac{\|\mathbf{g}(\mathbf{u}+\mathbf{e}^{(m)}) - \mathbf{g}(\mathbf{u})\|}{\|\mathbf{e}^{(m)}\|}.$$

If the iterative scheme converges, then $\mathbf{e}^{(m)} \to \mathbf{0}$ as $m \to \infty$. Given the hypothesis that $\mathbf{g}$ is a differentiable function, we see that the right side of the last equation approaches the norm of the Jacobian matrix of $\mathbf{g}$, evaluated at the solution $\mathbf{u}$, as $m \to \infty$:

$$\lim_{m\to\infty} \frac{\|\mathbf{e}^{(m+1)}\|}{\|\mathbf{e}^{(m)}\|} = \|\mathbf{g}'(\mathbf{u})\|. \tag{6.3-17}$$

Thus, at each iteration after the first few, we can expect the error to decrease roughly by a factor of $\|\mathbf{g}'(\mathbf{u})\|$.

Error-norm ratios such as the one estimated in Equation (6.3-17) provide the basis for the most common method of measuring the speed with which iterative schemes converge. We say that a scheme has **order of convergence** (or **asymptotic convergence rate**) α if there exists some constant C such that

$$\lim_{m\to\infty} \frac{\|\mathbf{e}^{(m+1)}\|}{\|\mathbf{e}^{(m)}\|^{\alpha}} \leq C.$$

We have just demonstrated that the method of successive substitution has order of convergence $\alpha = 1$, a fact that we commonly describe by saying that successive substitution converges linearly.

Nonlinear steady heat flow via successive substitution.

A similar approach using successive substitution applies to the equation

$$\nabla \cdot [K(u)\,\nabla u] = f(u),$$

only now the presence of a nonlinear coefficient $K(u)$ in the flux term on the left leads to some special considerations. Consider again a Galerkin finite-element formulation for a homogeneous Dirichlet problem forcing $u = 0$ on $\partial\Omega$. Employing the same trial function as for the nonlinear Poisson problem, we can derive the following Galerkin integral equations:

$$\int_\Omega [K(\hat{u})\,\nabla\hat{u}\cdot\nabla\phi_j + f(\hat{u})\phi_j]\,d\mathbf{x} = 0, \quad j = 1,\ldots,N.$$

Substituting the trial function given in Equation (6.3-11), we get

$$\sum_{i=1}^{N} \int_\Omega K(\hat{u})\,\nabla\phi_i\cdot\nabla\phi_j\,d\mathbf{x} + \int_\Omega f(\hat{u})\phi_j\,d\mathbf{x} = 0,$$

$$j = 1, \ldots, N.$$

This set of equations has an equivalent matrix form,

$$\mathbf{A}(\mathbf{u})\mathbf{u} = -\mathbf{f}(\mathbf{u}), \tag{6.3-18}$$

where the entries of the matrix $\mathbf{A}$ and the vector $\mathbf{f}$ are as follows:

$$A_{ji} = \int_\Omega K(\hat{u})\, \nabla\phi_i \cdot \nabla\phi_j \, d\mathbf{x}, \tag{6.3-19a}$$

$$f_j = \int_\Omega f(\hat{u})\phi_j \, d\mathbf{x}. \tag{6.3-19b}$$

Notice that, in contrast to the matrix equation (6.3-14) for the nonlinear Poisson problem, the forcing function *and* the matrix $\mathbf{A}$ in Equation (6.3-8) depend nonlinearly on the unknown solution $\mathbf{u}$. Given our previous development, this fact poses no real conceptual difficulty in formulating an iterative method. We can construct a successive substitution scheme simply by lagging the evaluation of $\mathbf{u}$ by an iteration whenever it appears as the argument of a function:

$$\mathbf{A}(\mathbf{u}^{(m)})\mathbf{u}^{(m+1)} = -\mathbf{f}(\mathbf{u}^{(m)}).$$

Each stage in this iterative procedure requires the solution of a linear matrix equation, and we can investigate the performance of the scheme as before if we examine the iterated function $\mathbf{g}(\mathbf{u}) = -\mathbf{A}^{-1}(\mathbf{u})f(\mathbf{u})$.

As a practical matter, however, the nonlinearities in the flux term now interfere with the computation of the matrix entries in $\mathbf{A}$. At the m-th iteration, the matrix entries given in Equation (6.3-19a) will look like

$$A_{ji}^{(m)} = \int_\Omega K(\hat{u}^{(m)})\, \nabla\phi_i \cdot \nabla\phi_j \, d\mathbf{x},$$

where $\hat{u}^{(m)}$ denotes the function obtained by substituting the coefficients stored in the most recently computed vector $\mathbf{u}^{(m)}$ into the trial function (6.3-11). These integrals may be difficult or impossible to compute exactly if the material property K has a complicated functional form.

One way to circumvent this problem is to use numerical quadrature, as discussed in Section 2.12. With this tactic, we would use the trial function $\hat{u}^{(m)}(\mathbf{x}) = \sum_{i=1}^N u_i^{(m)}\phi_i(\mathbf{x})$ to evaluate $K(\hat{u}^{(m)}(\mathbf{x}))$ at the requisite sampling points, multiply these values by the corresponding values of $\nabla\phi_i(\mathbf{x}) \cdot \nabla\phi_j(\mathbf{x})$, and add the results together in a weighted sum to compute an approximate value of each integral. Observe that this process requires a fair amount of calculation at every iteration, thereby contributing significantly to the cost of the computations.

A common alternative to this straightforward approach is to use functional coefficients, as introduced in Section 3.6. Thus, we might adopt a finite-element representation for K having the form

$$K(\hat{u}^{(m)}(\mathbf{x})) \simeq \hat{K}^{(m)}(\mathbf{x}) = \sum_{\ell=1}^{M} K_\ell^{(m)} \psi_\ell(\mathbf{x}).$$

Two observations are in order here. First, the basis functions ψ_ℓ used in this representation of K may be different from those used in the trial function $\hat{u}$. This flexibility may allow some computational savings in some formulations. For example, if $\hat{u}$ has a piecewise Hermite cubic expansion, it may be much simpler to compute a piecewise Lagrange linear expansion for K than to construct another Hermite cubic expansion for it. Second, the coefficients $K_\ell^{(m)}$ are simply the values of the function K at the values $\hat{u}^{(m)}(\mathbf{x}_\ell)$ corresponding to the appropriate spatial location in the expansion of $\hat{K}^{(m)}(\mathbf{x})$. If the nodes of the interpolating functions ψ_ℓ are also nodes of the interpolating functions ϕ_i, then we need not explicitly interpolate $\hat{u}^{(m)}$ to compute $\hat{K}^{(m)}$. Rather, we can simply set

$$K_\ell^{(m)} = K(u_\ell^{(m)}),$$

where $u_\ell^{(m)}$ signifies the nodal value corresponding to the node at which ψ_ℓ is centered.

To see how such an approximate scheme works computationally, notice that it yields approximate matrix entries having the form

$$\begin{aligned}(6.3-20)\quad A_{ji}^{(m)} &\simeq \int_\Omega \hat{K}^{(m)}(\mathbf{x})\, \nabla\phi_i(\mathbf{x}) \cdot \nabla\phi_j(\mathbf{x})\, d\mathbf{x} \\ &= \int_\Omega \left[\sum_{\ell=1}^{M} K(u_\ell^{(m)}\psi_\ell(\mathbf{x})\right] \nabla\phi_i(\mathbf{x}) \cdot \nabla\phi_j(\mathbf{x})\, d\mathbf{x} \\ &= \sum_{\ell=1}^{M} K(u_\ell^{(m)}) \int_\Omega \psi_\ell \nabla\phi_i \cdot \nabla\phi_j\, d\mathbf{x}.\end{aligned}$$

In light of the fact that each of the basis functions ψ_ℓ, ϕ_i, and ϕ_j vanishes over most of the finite-element grid, we see that very few of the integrals appearing in this sum will be nonzero. The sum will therefore be very easy to compute once we have calculated the appropriate coefficients $K(u_\ell^{(m)})$. More important, these integrals are independent of $\hat{u}$, and we can compute them once and for all in advance of starting the iterative procedure. The calculation of the nonlinear flux coefficient can now be limited to the nodes, and the evaluation of matrix entries amounts to the computation of

small linear combinations of these nodal values. As mentioned in Section 3.16, these linear combinations will be especially easy to compute if we simply choose piecewise constant basis functions ψ_ℓ for the finite-element representation of K.

Finally, we can adopt a similar functional representation to compute the forcing vector $\mathbf{f}(\mathbf{u})$ in the matrix equation (6.3-18). Using the same basis functions $\{\psi_\ell\}_{\ell=1}^M$ as for the coefficient K, we get

$$f(\hat{u}^{(m)}(\mathbf{x})) \simeq \hat{f}^{(m)}(\mathbf{x}) = \sum_{\ell=1}^{M} f_\ell^{(m)} \psi_\ell(\mathbf{x}),$$

where $f_\ell^{(m)} = f(u_\ell^{(m)})$. Thus we can approximate the j-th entry of the vector $\mathbf{f}$ as follows:

$$\begin{aligned} f_j &\simeq \int_\Omega \left[\sum_{\ell=1}^{M} f(u_\ell^{(m)}) \psi_\ell(\mathbf{x}) \right] \phi_j(\mathbf{x})\, d\mathbf{x} \\ &= \sum_{\ell=1}^{M} f(u_\ell^{(m)}) \int_\Omega \psi_\ell(\mathbf{x}) \phi_j(\mathbf{x})\, d\mathbf{x}. \end{aligned} \tag{6.3-21}$$

Once again, only a few of the integrals $\int_\Omega \psi_\ell \phi_j d\mathbf{x}$ will be nonzero, and those that are can be computed in advance of the iterations. The computation of the forcing vector thus also reduces to the calculation of small linear combinations of function values at each iteration.

Nonlinear steady heat flow via Newton's method.

Another class of iterative approaches for discretized nonlinear PDEs uses **Newton's method** as its point of departure. If we write the discrete equation (6.3-18) for the coefficient vector $\mathbf{u} = (u_1, \ldots, u_N)$ in the form

$$\mathbf{F}(\mathbf{u}) = \mathbf{A}(\mathbf{u})\mathbf{u} + \mathbf{f}(\mathbf{u}) = \mathbf{0}, \tag{6.3-22}$$

then it becomes apparent that solving the finite-element equations is equivalent to finding a root of the nonlinear vector function $\mathbf{F}$. Newton's method for an equation of the form $\mathbf{F}(\mathbf{u}) = \mathbf{0}$ is an iterative scheme based on the following observation. If $\mathbf{F}$ is twice continuously differentiable as a function of the vector $\mathbf{u}$, then it has a Taylor expansion about any known iterative value $\mathbf{u}^{(m)}$:

$$\mathbf{F}(\mathbf{u}^{(m)} + \Delta\mathbf{u}) = \mathbf{F}(\mathbf{u}^{(m)}) + \mathbf{F}'(\mathbf{u}^{(m)})\Delta\mathbf{u} + \mathcal{O}(\|\Delta\mathbf{u}\|_2^2). \tag{6.3-23}$$

Here, $\Delta\mathbf{u}$ is an increment in the value of $\mathbf{u}^{(m)}$. $\mathbf{F}'(\mathbf{u})$ denotes the Jacobian matrix of $\mathbf{F}(\mathbf{u})$, whose (j,i)-th entry is $\partial F_j / \partial u_i$, F_j being the j-th component function of $\mathbf{F}(\mathbf{u}) = (F_1(\mathbf{u}), \ldots, F_N(\mathbf{u}))$.

Given an iterative value $\mathbf{u}^{(m)}$, if we could find an increment $\Delta\mathbf{u}$ such that $\mathbf{F}(\mathbf{u}^{(m)} + \Delta\mathbf{u}) = \mathbf{0}$, then we would be able to solve Equation (6.3-22) exactly. However, the nonlinearity of $\mathbf{F}$ usually prevents this. Let us settle instead on neglecting the term $\mathcal{O}(\|\Delta\mathbf{u}\|_2^2)$ in Equation (6.3-23) and forcing the resulting *linear* approximation $\mathbf{F}(\mathbf{u}^{(m)}) + \mathbf{F}'(\mathbf{u}^{(m)})\Delta\mathbf{u}$ to vanish:

$$\mathbf{F}'(\mathbf{u}^{(m)})\Delta\mathbf{u} = -\mathbf{F}(\mathbf{u}^{(m)}). \tag{6.3-24a}$$

Since $\mathbf{u}^{(m)}$ is a known vector, the entries of the matrix $\mathbf{F}'(\mathbf{u}^{(m)})$ and the vector $\mathbf{F}(\mathbf{u}^{(m)})$ can be computed to give a linear matrix equation for the increment vector $\Delta\mathbf{u}$. We then form a new iterate $\mathbf{u}^{(m+1)}$ by setting

$$\mathbf{u}^{(m+1)} = \mathbf{u}^{(m)} + \Delta\mathbf{u}. \tag{6.3-24b}$$

While this new iterate will generally fail to solve the nonlinear problem exactly, we expect that it will furnish a better approximation to the true solution than the previous iterate. Having computed the new iterate, we can go on to compute yet another new iterate, proceeding in this fashion until the residual $\mathbf{F}(\mathbf{u}^{(m+1)})$ is small enough in some norm. Equations (6.3-24) constitute Newton's method for solving $\mathbf{F}(\mathbf{u}) = \mathbf{0}$ iteratively.

Before examining the application of this scheme to our discretized PDE, let us see what to expect for its convergence rate. Our argument will rest on the assumption that the Jacobian matrix $\mathbf{F}'$ is nonsingular throughout some neighborhood of the exact root $\mathbf{u}$. Provided $\mathbf{F}$ is sufficiently smooth, we can use Taylor's theorem to expand $\mathbf{F}$ at its exact root $\mathbf{u}$ about the known iterate $\mathbf{u}^{(m)}$ as follows:

$$\mathbf{0} = \mathbf{F}(\mathbf{u}) = \mathbf{F}(\mathbf{u}^{(m)}) + \mathbf{F}'(\mathbf{u}^{(m)})(\mathbf{u} - \mathbf{u}^{(m)}) + \mathbf{R}.$$

In this expansion the remainder $\mathbf{R} = \mathcal{O}(\|\mathbf{u} - \mathbf{u}^{(m)}\|^2)$, meaning that $\|\mathbf{R}\|/\|\mathbf{u} - \mathbf{u}^{(m)}\|^2 \leq C$ for some constant C as $m \to \infty$. Now notice that $\mathbf{u} - \mathbf{u}^{(m)} = -\mathbf{e}^{(m)}$, where $\mathbf{e}^{(m)}$ is the error at the m-th iteration, and that $\mathbf{F}(\mathbf{u}^{(m)}) = -\mathbf{F}'(\mathbf{u}^{(m)})(\mathbf{u}^{(m+1)} - \mathbf{u}^{(m)})$ by the definition of Newton's method. Making these substitutions, we find

$$\mathbf{F}'(\mathbf{u}^{(m+1)} - \mathbf{u}) = \mathbf{R},$$

and

$$\frac{\|\mathbf{R}\|}{\|\mathbf{e}^{(m)}\|^2} \to C \quad \text{as} \quad m \to \infty.$$

When Newton's method converges, $\mathbf{F}'(\mathbf{u}^{(m)}) \to \mathbf{F}'(\mathbf{u})$ as $m \to \infty$, and, since $\mathbf{u}^{(m+1)} - \mathbf{u} = \mathbf{e}^{(m+1)}$, taking norms yields

$$\frac{\|\mathbf{e}^{(m+1)}\|}{\|\mathbf{e}^{(m)}\|^2} \leq \frac{\|\mathbf{R}\|}{\|\mathbf{e}^{(m)}\|^2} \frac{1}{\|\mathbf{F}'(\mathbf{u}^{(m)})\|} \leq \frac{C}{\|\mathbf{F}'(\mathbf{u})\|}$$

$$\text{as} \quad m \to \infty.$$

Since the right side of this limit is a constant, we see that the order of Newton's method is two provided our assumptions about $\mathbf{F}$ hold. In other words, Newton's method converges quadratically.

Now let us apply the method to Equation (6.3-22). To make the development both simple and realistic, consider the numerical approximation to this equation that results when we adopt functional representations of the forms (6.3-20) and (6.3-21) for the entries of $\mathbf{A}$ and $\mathbf{f}$. Thus

$$A_{jn}(\mathbf{u}) \simeq \sum_{\ell=1}^{M} \alpha_{nj\ell} K(u_\ell),$$

$$f_j(\mathbf{u}) \simeq \sum_{\ell=1}^{M} \beta_{j\ell} f(u_\ell),$$

where $\alpha_{nj\ell} = \int_\Omega \psi_\ell \nabla\phi_n \cdot \nabla\phi_j \, d\mathbf{x}$ and $\beta_{j\ell} = \int_\Omega \psi_\ell \phi_j \, d\mathbf{x}$ are constants and u_ℓ stands for an entry in the sought solution vector $\mathbf{u}$. Therefore the j-th component of the function $\mathbf{F}$ takes the form

$$\begin{aligned} F_j(\mathbf{u}) &= \sum_{n=1}^{N} A_{jn} u_n + f_j(\mathbf{u}) \\ &\simeq \sum_{n=1}^{N} \left[\sum_{\ell=1}^{M} \alpha_{nj\ell} K(u_\ell) \right] u_n + \sum_{\ell=1}^{M} \beta_{j\ell} f(u_\ell). \end{aligned}$$

We can thus compute the (j,i)-th entry of the Jacobian matrix for $\mathbf{F}$ approximately as follows:

$$\begin{aligned} \frac{\partial F_j}{\partial u_i}(\mathbf{u}^{(m)}) &\simeq \sum_{n=1}^{N} \alpha_{nji} K'(u_i^{(m)}) u_n^{(m)} \\ &\quad + \sum_{\ell=1}^{M} \alpha_{ij\ell} K(u_\ell^{(m)}) + \beta_{ji} f'(u_i^{(m)}). \end{aligned} \tag{6.3-25}$$

Observe that, even though each of these entries is a relatively small linear combination, each requires the evaluation of the nonlinear functions $K(u)$, $K'(u)$, and $f'(u)$ at several values of their arguments. What is more, we must recompute each entry in the $N \times N$ Jacobian matrix $\mathbf{F}'$ and the forcing vector $-\mathbf{F}$ at every iteration. These calculations can contribute significantly to the computational cost of Newton's method, especially if, as often happens, the partial derivatives of $\mathbf{F}$ require more effort to compute than $\mathbf{F}$ itself.

There are many variants on Newton's method aimed at reducing this computational cost. One variant that is quite simple is to use the initial Jacobian matrix throughout the iterations. Thus, instead of Equations (6.3-24), we use

$$\mathbf{F}'(\mathbf{u}^{(0)})\,\Delta\mathbf{u} = -\mathbf{F}(\mathbf{u}^{(m)}),$$
$$\mathbf{u}^{(m+1)} = \mathbf{u}^{(m)} + \Delta\mathbf{u}.$$

This scheme, called the **modified Newton's method**, clearly relies on the assumption that the Jacobian matrix $\mathbf{F}'$ varies slowly as a function of $\mathbf{u}$ in some neighborhood of the exact solution. The method typically exhibits slower convergence, but in many problems the savings per iteration gained by avoiding the construction and inversion of a new Jacobian matrix can outweigh the cost of the extra iterations required.

Another class of variants on Newton's method bypasses the computation of derivatives altogether. In many practical problems the functions K and f may not be amenable to differentiation in closed form. In these cases Equation (6.3-25) does not provide a computable expression for $\partial F_i/\partial u_i$. One alternative is to replace this partial derivative by a finite-difference approximation:

$$\frac{\partial F_j}{\partial u_i}(\mathbf{u}^{(m)}) \simeq \frac{1}{h_i^{(m)}}\left[F_j(\mathbf{u}^{(m)} + h_i^{(m)}\mathbf{e}_i) - F_j(\mathbf{u}^{(m)})\right],$$

where $\mathbf{e}_i$ denotes the i-th unit basis vector whose j-th entry is the Kronecker symbol δ_{ij} and $h_i^{(m)}$ signifies an increment chosen in some systematic way. We shall review two ways of choosing this increment, referring the reader to Ortega and Rheinboldt (1970) for a thorough discussion of the various possibilities as well as for proofs of the convergence rates stated below.

One seemingly natural choice for the increment $h_i^{(m)}$ is $h_i^{(m)} = u_i^{(m)} - u_i^{(m-1)}$, whose magnitude presumably decreases as $m \to \infty$. Obviously, this choice makes sense only if $u_i^{(m)} \neq u_i^{(m-1)}$. The resulting scheme, known as the **secant method**, requires $N^2 + 1$ evaluations of the component functions F_j at each iteration, and its order of convergence is $(1 + \sqrt{5})/2 \simeq 1.618$. A somewhat more sophisticated approach is **Steffensen's method**, in which we choose $h_i^{(m)} = F_i(\mathbf{u}^{(m)})$, which should also decrease in magnitude as $k \to \infty$. When discrepancies in the units of $h_i^{(m)}$ and $F_i(\mathbf{u}^{(m)})$ arise, or when $F_i(\mathbf{u}^{(m)})$ has an inappropriate magnitude, we can multiply the latter by a suitable scaling factor. Steffensen's method offers the advantage of retaining the quadratic convergence rate associated with Newton's method while obviating the computation of derivatives.

One final Newton-like method that we shall examine is based on the original PDE rather than its discrete analog. Let us write the steady heat-flow problem as follows:

$$\mathcal{F}(u) \equiv \nabla \cdot [K(u)\nabla u] - f(u) = 0.$$

We can derive a linearized form of this equation by evaluating the function $K(u)$ at a known iterative level m and linearly projecting the remaining occurrences of $u(\mathbf{x})$ to the next iterative level:

$$\nabla \cdot [K(u^{(m)})\nabla u^{(m+1)}] - f(u^{(m+1)}) = 0.$$

Now we rewrite this equation so that the increment $\delta u = u^{(m+1)} - u^{(m)}$ appears as the unknown. Replacing $u^{(m+1)}$ by $u^{(m)} + \delta u$, adopting the approximation $f(u^{(m+1)}) \simeq f(u^{(m)}) + f'(u^{(m)})\delta u$, and rearranging, we get

$$\left\{\nabla \cdot [K(u^{(m)})\nabla] - f'(u^{(m)})\right\}\delta u \tag{6.3-26}$$

$$= -\left\{\nabla \cdot [K(u^{(m)})\nabla u^{(m)}] - f(u^{(m)})\right\} = -\mathcal{F}(u^{(m)}).$$

This scheme furnishes a linear operator equation that we can solve for the iterative increment δu, after which we set $u^{(m+1)} = u^{(m)} + \delta u$ to begin a new iteration. Notice the formal similarity between Equation (6.3-26) and the matrix equation (6.3-24a) defining Newton's method: An operator evaluated at the most recently known iterative level acts on an unknown increment to yield the negative of the most recent residual.

To discretize Equation (6.3-26), we can establish, for example, a finite-element representation

$$\hat{u}^{(m)}(\mathbf{x}) = u_\partial(\mathbf{x}) + \sum_{i=1}^{N} u_i^{(m)} \phi_i(\mathbf{x}), \tag{6.3-27}$$

where $u_\partial(\mathbf{x})$ satisfies the known essential boundary conditions and the basis functions $\phi_i(\mathbf{x})$ satisfy homogeneous boundary conditions. We then use an analogous trial function for the unknown increment δu:

$$\delta\hat{u}(\mathbf{x}) = \sum_{i=1}^{N} \delta u_i \, \phi_i(\mathbf{x}).$$

If we substitute these expansions into the operator equation (6.3-27) and apply some version of the method of weighted residuals, we shall produce a matrix analog of Equation (6.3-26) that is linear in the vector $(\delta u_1, \ldots, \delta u_N)$ of unknown nodal increments. We then solve this matrix equation at each iteration, using the new increment vectors to update the coefficients in Equation (6.3-27) according to the relationship $u_i^{(m+1)} = u_i^{(m)} + \delta u_i$ at each iterative level. As with Newton's method, we can stop iterating as soon as the residual $\mathcal{F}(\hat{u}^{(m)})$ is small enough in some norm.

The nonlinear diffusion equation.

We now turn our attention to the nonlinear diffusion equation

$$\frac{\partial}{\partial t}[c(u)] - \nabla \cdot [K(u)\nabla u] = 0. \tag{6.3-9}$$

Two features distinguish this equation from the nonlinear steady heat-flow problem just considered. First, we must discretize the equation in both space and time, so the issue of time-stepping algorithms becomes crucial. Second, Equation (6.3-9) exhibits nonlinearity in the accumulation term $\partial[c(u)]/\partial t$. In the context of the species mass balance from which we derived Equation (6.3-9), the nonlinearity in the accumulation term has a common physical interpretation in terms of compositional density effects. Indeed, if we apply the chain rule to the time derivative of the solute density $c(u)$, we find

$$\frac{\partial}{\partial t}[c(u)] = \frac{dc}{du}(u)\frac{\partial u}{\partial t}. \tag{6.3-28}$$

The function dc/du represents the rate of change of density with respect to solute mass fraction u, which we might identify as a "compressibility" due to the effects of mixture composition.

It is quite common to discretize Equation (6.3-9) in time by first applying the chain rule as in Equation (6.3-28) and then approximating the quantity $\partial u/\partial t$ by finite differences or some other method. Thus we obtain a **semidiscrete** equation whose form depends on which time level we choose for the evaluations of the remaining occurrences of u in the PDE. If the index n signifies the most recent known time level, then evaluating the spatial terms and the coefficient dc/du at the next unknown time level $n+1$ produces an implicit equation,

$$\frac{1}{k}\frac{dc}{du}(u^{n+1})(u^{n+1} - u^n) - \nabla \cdot \left[K(u^{n+1})\nabla u^{n+1}\right] = 0,$$

where k denotes the time step.

At this point we can choose any one of several schemes to discretize the spatial variations. Let us examine the application of finite-element collocation, as discussed in Section 2.14. For simplicity, we shall treat the one-dimensional analog,

$$\frac{1}{k}\frac{dc}{du}(u^{n+1} - u^n) - \frac{\partial}{\partial x}\left[K(u^{n+1})\frac{\partial u^{n+1}}{\partial x}\right] = 0, \tag{6.3-29}$$

on a spatial interval $(0, L)$. Let us consider this equation subject to the mixed, constant boundary conditions

$$u(0,t) = \bar{u}_0, \quad t > 0,$$
$$\frac{\partial u}{\partial x}(L,t) = \bar{u}'_L, \quad t > 0,$$

and initial conditions

$$u(x,0) = u_I(x), \quad x \in (0,L).$$

For the finite-element basis functions we select the piecewise Hermite cubic interpolating polynomials $\{h_i^0(x), h_i^1(x)\}_{i=0}^N$ on a uniform grid $\Delta : 0 = x_0 < x_1 < \cdots < x_N = L$ having mesh $x_i - x_{i-1} = h$. For the given boundary data, our trial function at time level n will have the form

$$\begin{aligned}\hat{u}^n(x) = {} & \bar{u}_0 h_0^0(x) + (u'_0)^n h_0^1(x) \\ & + \sum_{i=1}^{N-1} \left[(u_i)^n h_i^0(x) + (u'_i)^n h_i^1(x) \right] \\ & + (u_n)^n h_N^0(x) + \bar{u}'_L h_N^1(x),\end{aligned} \tag{6.3-30}$$

where the coefficients $(u_1)^n, \ldots, (u_N)^n$ stand for unknown values of $\hat{u}^n(x)$ at the nodes $x_1, \ldots, x_N$, and the coefficients $(u'_0)^n, \ldots, (u'_{N-1})^n$ stand for unknown values of $d\hat{u}^n/dx$ at the nodes $x_0, \ldots, x_{N-1}$. Thus at each time level $t = nk$ we must solve for $2N$ unknown coefficients defining the trial function $\hat{u}^n \in C^1((0,L))$.

To get the necessary $2N$ equations, we collocate Equation (6.3-29) at the $2N$ Gauss points, as discussed in Section 2.14. Let us denote these points as $\bar{x}_\ell$, $\ell = 1, \ldots, 2N$. Recall that these points, when viewed in the local coordinates ξ mapping a typical element $[x_{i-1}, x_i]$ onto $[-1, 1]$, lie at $\xi(\bar{x}_\ell) = \pm 1/\sqrt{3} \simeq \pm 0.57735$. Using these points, we arrive at the equations

$$\begin{aligned}&\frac{1}{k}\frac{dc}{du}(\hat{u}^{n+1}(\bar{x}_\ell))\left[\hat{u}^{n+1}(\bar{x}_\ell) - \hat{u}^n(\bar{x}_\ell)\right] \\ &- \frac{\partial}{\partial x}\left[K(\hat{u}^{n+1}(\bar{x}_\ell))\frac{\partial \hat{u}^{n+1}}{\partial x}(\bar{x}_\ell)\right] = 0, \quad \ell = 1, \ldots, 2N,\end{aligned}$$

or, equivalently,

$$\begin{aligned}&\frac{1}{k}\frac{dc}{du}(\hat{u}^{n+1}(\bar{x}_\ell))\left[\hat{u}^{n+1}(\bar{x}_\ell) - \hat{u}^n(\bar{x}_\ell)\right] \\ &- \frac{\partial K}{\partial x}(\hat{u}^{n+1}(\bar{x}_\ell))\frac{\partial \hat{u}^{n+1}}{\partial x}(\bar{x}_\ell) - K(\hat{u}^{n+1}(\bar{x}_\ell))\frac{\partial^2 \hat{u}^{n+1}}{\partial x^2}(\bar{x}_\ell) \\ &= 0, \qquad \ell = 1, \ldots, 2N.\end{aligned} \tag{6.3-31}$$

We now confront the fact that the nonlinear functions dc/du, K, and $\partial K/\partial x$ make the discrete equations impossible to solve, except perhaps in very simple cases. To circumvent this difficulty, we must devise convenient approximations for these coefficients and implement an iterative scheme to accommodate their dependence on the unknown $\hat{u}^{n+1}$. For the first task, let us approximate dc/du and K by functional representations using piecewise Lagrange linear expansions:

$$\frac{dc}{du}(\hat{u}^{n+1}) \simeq \widehat{\left(\frac{dc}{du}\right)}^{n+1}(x) = \sum_{i=0}^{N} \left(\frac{dc}{du}\right)_i^{n+1} \ell_i(x),$$

$$K(\hat{u}^{n+1}) \simeq \hat{K}^{n+1}(x) = \sum_{i=0}^{N} K_i^{n+1} \ell_i(x).$$

In these equations, $\ell_i(x)$ denotes the piecewise Lagrange linear basis function associated with the node x_i, and the nodal values $(dc/du)_i^{n+1}$ and K_i^{n+1} are just the values of the functions dc/du and K at the nodal values of $\hat{u}^{n+1}$:

$$\left(\frac{dc}{du}\right)_i^{n+1} = \frac{dc}{du}((u_i)^{n+1});$$

$$K_i^{n+1} = K((u_i)^{n+1}).$$

Observe that this representation for K suggests a natural representation for $\partial K/\partial x$, namely,

$$\frac{\partial K}{\partial x}(\hat{u}^{n+1}) \simeq \frac{\partial \hat{K}}{\partial x}(x) = \sum_{i=0}^{N} K_i^{n+1} \frac{d\ell_i}{dx}(x).$$

We are left with the job of constructing an iterative scheme to handle the nonlinearities. Following our discussion of the nonlinear steady heat-flow equation, we shall employ a Newton-like scheme in which, given a known iterative value $\hat{u}^{n+1,m}(x)$ for the new time level $n+1$, we solve for an increment

$$\delta\hat{u}(x) = \sum_{i=1}^{N} \left[\delta u_i \, h_i^0(x) + \delta u_i' \, h_i^1(x)\right]$$

and compute the coefficients of the new iterate $\hat{u}^{n+1,m+1}(x)$ according to the updating rules

$$(u_i)^{n+1,m+1} = (u_i)^{n+1,m} + \delta u_i,$$

$$(u_i')^{n+1,m+1} = (u_i')^{n+1,m} + \delta u_i'.$$

Observe that the boundary values $(u_0)^{n+1} = \bar{u}_0$ and $(u'_N)^{n+1} = \bar{u}'_L$ are known, so $\delta u_0 = \delta u'_N = 0$ at every stage.

In solving for these increments, we can lag all nonlinear coefficients by an iteration, projecting linear occurrences of the unknown $\hat{u}^{n+1}$ forward to the next unknown iterative level. We thus arrive at a linearized set of collocation equations,

$$\begin{aligned}
&\frac{1}{k}\widehat{\left(\frac{dc}{du}\right)}^{n+1,m}(\bar{x}_\ell)\left[\hat{u}^{n+1,m}(\bar{x}_\ell) + \delta\hat{u}(\bar{x}_\ell) - \hat{u}^n(\bar{x}_\ell)\right] \\
&- \frac{\partial \hat{K}^{n+1,m}}{\partial x}(\bar{x}_\ell)\left[\frac{\partial \hat{u}^{n+1,m}}{\partial x}(\bar{x}_\ell) + \frac{\partial \delta\hat{u}}{\partial x}(\bar{x}_\ell)\right] \\
&- \hat{K}^{n+1,m}(\bar{x}_\ell)\left[\frac{\partial^2 \hat{u}^{n+1,m}}{\partial x^2}(\bar{x}_\ell) + \frac{\partial^2 \delta\hat{u}}{\partial x^2}(\bar{x}_\ell)\right] = 0, \\
&\qquad \ell = 1, \ldots, 2N.
\end{aligned}$$

If we move all terms known at the m-th iterative level to the right side of this equation and keep the terms involving the unknown coefficients of $\delta\hat{u}$ on the left, we get

$$\begin{aligned}
(6.3\text{-}32)\quad &\left[\frac{1}{k}\widehat{\left(\frac{dc}{du}\right)}^{n+1,m}(\bar{x}_\ell) - \frac{\partial \hat{K}^{n+1,m}}{\partial x}(\bar{x}_\ell)\frac{\partial}{\partial x} - \hat{K}^{n+1,m}(\bar{x}_\ell)\frac{\partial^2}{\partial x^2}\right]\delta\hat{u}(\bar{x}_\ell) \\
&= -\left\{\frac{1}{k}\widehat{\left(\frac{dc}{du}\right)}^{n+1,m}(\bar{x}_\ell)\left[\hat{u}^{n+1,m}(\bar{x}_\ell) - \hat{u}^n(\bar{x}_\ell)\right]\right. \\
&\left. - \frac{\partial \hat{K}^{n+1,m}}{\partial x}(\bar{x}_\ell)\frac{\partial \hat{u}^{n+1,m}}{\partial x}(\bar{x}_\ell) - \hat{K}^{n+1,m}(\bar{x}_\ell)\frac{\partial^2 \hat{u}^{n+1,m}}{\partial x^2}(\bar{x}_\ell)\right\} \\
&\equiv -\hat{R}^{n+1,m}(\bar{x}_\ell), \quad \ell = 1, \ldots, 2N.
\end{aligned}$$

We terminate the iterative calculations for each new time level $n+1$ as soon as the residual norm $\|\hat{R}^{n+1,m+1}\|_\infty$ falls below some prescribed tolerance, setting $\hat{u}^{n+1} = \hat{u}^{n+1,m+1}$. Given the computed value $\hat{u}^n(x)$ at any time level, we can begin the iterations for the next time level by setting $\hat{u}^{n+1,0} = \hat{u}^n$. We leave to the reader the task of showing that each iteration involves the solution of a set of linear equations whose matrix has the structure drawn in Figure 6-4.

Although the discrete equations (6.3-32) are consistent with the original PDE (6.3-12), they tend to produce approximate solutions having global mass balance errors. More precisely, unless the time step k is very small or the coefficient dc/du varies slowly as a function of the unknown u, the mass error over a single time step can be significant. The integral

$$\int_0^L \left[c(\hat{u}^{n+1}) - c(\hat{u}^n) - k\frac{\partial}{\partial x}\left(\hat{K}^{n+1}\frac{\partial \hat{u}^{n+1}}{\partial x}\right)\right] dx$$

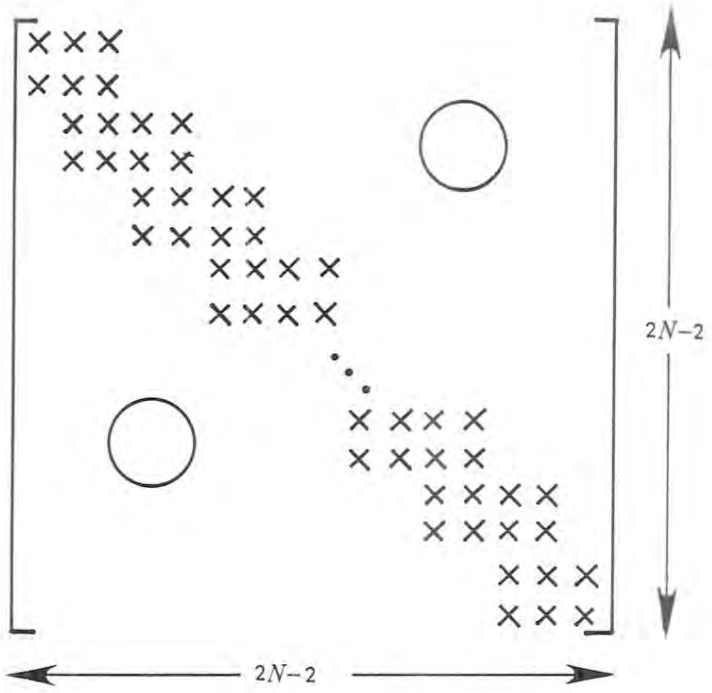

Figure 6-4. Structure of the matrix arising at each iteration in the finite-element collocation scheme for solving the nonlinear diffusion equation.

furnishes a computable measure of this error. The problem with Equations (6.3-32) is their treatment of the accumulation term: The value of the coefficient $(\widehat{dc/du})$ at the level $(n+1,m)$ may fail to be representative of its value over the entire time step from $t = nk$ to $t = (n+1)k$. To avoid the thorny issue of how to evaluate this coefficient more accurately, it is sometimes best to abandon the convenience of the chain-rule factorization in Equation (6.3-28) in favor of a *direct* discretization of the accumulation term. Thus we need a functional representation for the constitutive function $c(u)$, which, for consistency with the Hermite cubic representation used to compute the spatial flux terms, we can take to be

$$\hat{c}^{n+1} = \sum_{i=0}^{N} \left[c((u_i)^{n+1}) h_i^0(x) + \frac{dc}{du}((u_i)^{n+1})(u_i')^{n+1} h_i^1(x) \right].$$

Notice that we have used the chain rule to develop a representation for the slope $\partial \hat{c}/\partial x$ at each spatial node.

Using this approach, one can discretize the accumulation term in the iterative time-stepping scheme as follows:

$$\left.\frac{\partial \hat{c}}{\partial t}\right|^{n+1,m} \simeq \frac{1}{k}\left[\hat{c}^{n+1,m} + \left(\widehat{\frac{dc}{du}}\right)^{n+1,m} \delta\hat{u} - \hat{c}^{n} \right],$$

where we use the piecewise linear representation for $(\widehat{dc/du})$ as before to avoid polynomial approximations having large degree. The collocation equations that result are similar to those in Equation (6.3-32), the only exception being that

$$\hat{R}^{n+1,m}(\bar{x}_\ell) = \frac{1}{k}\left[\hat{c}^{n+1,m}(\bar{x}_\ell) - \hat{c}^{n}(\bar{x}_\ell)\right]$$
$$- \frac{\partial \hat{K}^{n+1,m}}{\partial x}(\bar{x}_\ell)\frac{\partial \hat{u}^{n+1,m}}{\partial x}(\bar{x}_\ell) - \hat{K}^{n+1,m}(\bar{x}_\ell)\frac{\partial^2 \hat{u}^{n+1,m}}{\partial x^2}(\bar{x}_\ell).$$

Now, when $\|\hat{R}^{n+1,m+1}\|_\infty$ is small, the integrated mass balance error over the time step will be small as well. For an application of this technique to a problem with strong nonlinearities in the accumulation term, we refer the reader to Allen and Murphy (1985). A subsequent paper (Allen and Murphy, 1986) demonstrates the extension of this approach to two space dimensions.

6.4. The Simulation of Solid Deformation.

As our first example of a coupled system of PDEs, we shall consider the problem of a fluid-saturated porous medium that undergoes deformation

as a result of fluid withdrawal. This system has great importance in areas where groundwater withdrawal leads to land subsidence. The numerical method of choice for the simulation of solid deformation is the finite-element method. The reason for this is the relative ease with which the method accommodates the moving boundaries of the porous medium as it deforms. Our development in this section follows that presented in Safai (1977).

Governing equations.

The governing equations describing soil deformation are obtained by substituting suitable constitutive relationships into the mixture balance laws appropriate for a fluid-saturated porous medium. Let us consider first the momentum balance for the fluid-solid mixture, recalling the framework for deriving multiphase balance laws presented in Section 1.5. If we denote the rock phase by the index R and the fluid phase by the index F, then summing the momentum balances for the two phases R and F yields

$$\rho \frac{D\mathbf{v}}{Dt} - \nabla \cdot (\mathbf{t}^R + \mathbf{t}^F) - \rho \mathbf{b} = \mathbf{0}.$$

Here, ρ stands for the overall mixture density, $\mathbf{v}$ is the barycentric velocity of the mixture, and $\mathbf{t}^R$ and $\mathbf{t}^F$ stand for the stress tensors in the solid and fluid, respectively. The symbol $\mathbf{b}$ signifies the total body force acting on the mixture, which we assume to be accounted for by the influence of gravity. In the most sophisticated models of solid displacement, every term in this equation may contribute to the net motion of the mixture. However, in many applications the motion resulting from changes in the stress term dominates the effects of the mixture's inertia and the influence of gravity. If we neglect these latter terms, we are left with

$$\nabla \cdot (\mathbf{t}^R + \mathbf{t}^F) = \mathbf{0}.$$

In Section 1.5 we argued that the stress in the fluid phase can be approximated by its isotropic part $-p^F \mathbf{1}$, where p^F is the mechanical pressure in the fluid. Adopting this approximation, we see that the overall momentum balance reduces to the following:

$$\nabla \cdot \mathbf{t}^R - \nabla \cdot (p^F \mathbf{1}) = \mathbf{0}. \tag{6.4-1}$$

Hydrologists working with solid deformation problems often call the quantities $\mathbf{t}^R$ and p^F the **effective stress** and the **excess pore-water pressure**, respectively.

Now let us examine the mass balances for the two phases. Before writing these equations, let us establish some assumptions and terminology.

In the following, we shall assume that the fluid and solid are incompressible in the sense that their intrinsic mass densities ρ^F and ρ^R have negligible rates of change in time and space, even though changes in the volume fractions $\phi^F \equiv \phi$ and $\phi^R = 1 - \phi$ may cause the *overall* mixture density to vary. We shall also refer to the fluid and solid displacements $\mathbf{U}^F$ and $\mathbf{U}^R$. We define each of these quantities, as for the solid displacement discussed in Section 1.4, as the distance between a material point in the appropriate phase and its location in the initial configuration of the mixture. Thus we may write the velocity in each phase α as $\mathbf{v}^\alpha = \partial \mathbf{U}^\alpha / \partial t$. Finally, recall that in Section 1.5 we derived Darcy's law for the velocity of a fluid in a porous medium by assuming the rock matrix to be stationary. In the problem of solid deformation we must abandon that assumption, since the solid phase will be moving. However, we can easily accommodate solid motions by referring to the **relative velocity** $\mathbf{v}^F - \mathbf{v}^R$ in the formulation of the Stokes drag of Section 1.5. The result is the following generalization of Darcy's law:

$$\mathbf{v}^F - \mathbf{v}^R \equiv \frac{\partial \mathbf{U}^F}{\partial t} - \frac{\partial \mathbf{U}^R}{\partial t} = -\frac{K}{\phi \rho^F g} \nabla p^F.$$

Observe that we have assumed gravitational forces to be negligible and have adopted the hydrologists' notation in using the hydraulic conductivity K, defined in Section 3.1. We use g to denote the magnitude of gravitational acceleration.

With these remarks in mind, consider the fluid mass balance,

$$\frac{\partial}{\partial t}(\phi \rho^F) + \nabla \cdot (\phi \rho^F \mathbf{v}^F) = 0.$$

Let us expand the time derivative and use the **fluid compressibility**, defined as $\beta = (1/\rho^F) d\rho^F / dp^F$, to rewrite the time derivative of the fluid density. We get

$$\phi \beta \rho^F \frac{\partial p^F}{\partial t} + \rho^F \frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \rho^F \mathbf{v}^F) = 0.$$

Next, notice that, according to Darcy's law, we can replace the fluid velocity in this last equation by using the identity

$$\mathbf{v}^F = -\frac{K}{\phi \rho^F g} \nabla p^F + \mathbf{v}^R.$$

Then if we use the assumption that gradients of ρ^F are negligible and further assume that the porous medium has uniform hydraulic conductivity, we get the following flow equation for the fluid:

$$\phi \beta \frac{\partial p^F}{\partial t} + \frac{\partial \phi}{\partial t} - \frac{K}{\rho^F g} \nabla^2 p^F + \nabla \cdot (\phi \mathbf{v}^R) = 0.$$

Turning to the mass balance for the solid matrix, we have

$$\frac{\partial}{\partial t}[(1-\phi)\rho^R] + \nabla \cdot [(1-\phi)\rho^R \mathbf{v}^R] = 0.$$

Now we expand the time derivative as before, this time assuming that the solid grains of the porous matrix are incompressible so that $\partial \rho^R / \partial t = 0$. We have as a result

$$\begin{aligned} -\frac{\partial \phi}{\partial t} &= -\nabla \cdot [(1-\phi)\mathbf{v}^R] \\ &= -\nabla \cdot \mathbf{v}^R + \nabla \cdot (\phi \mathbf{v}^R) \\ &= -\nabla \cdot \left(\frac{\partial \mathbf{U}^R}{\partial t} \right) + \nabla \cdot (\phi \mathbf{v}^R), \end{aligned}$$

where we have taken advantage of the spatial uniformity of ρ^R. Adding this equation to the flow equation for the fluid derived above and interchanging spatial and temporal differential operators, we obtain the final form of the flow equation,

$$\phi\beta \frac{\partial p^F}{\partial t} - \frac{K}{\rho^F g} \nabla^2 p^F + \frac{\partial}{\partial t}(\nabla \cdot \mathbf{U}^R) = 0. \tag{6.4-2}$$

In what follows, we shall consider the solid matrix to be a linear elastic solid. Thus we can rewrite the term $\nabla \cdot \mathbf{t}^R$ in Equation (6.4-1) using Hooke's law, Equation (1.4-12), which states

$$\mathbf{t}^R = 2\mu \mathbf{e}^R + \lambda \mathrm{tr}(\mathbf{e})\mathbf{1}, \tag{6.4-3}$$

where λ and μ are the Lamé constants of the rock matrix and $\mathbf{e} = [\nabla \mathbf{U}^R + (\nabla \mathbf{U}^R)^\top]$ is the infinitesimal Eulerian strain. Substituting this constitutive relationship into Equation (6.4-1) yields

$$\mu \nabla^2 \mathbf{U}^R + (\lambda + \mu) \nabla (\nabla \cdot \mathbf{U}^R) - \nabla \cdot (p^F \mathbf{1}) = \mathbf{0}. \tag{6.4-4}$$

To keep the notation from becoming too cumbersome, in the remainder of this section we shall drop the phase indices F and R from the governing equations.

Galerkin formulation.

Let us discretize Equations (6.4-1) [or (6.4-4)] and (6.4-2), treating the displacement vector $\mathbf{U}$ and the excess pore-water pressure p as the dependent variables. Let the approximating trial functions for these variables be defined as

$$\hat{\mathbf{U}}(\mathbf{x}, t) = \sum_{i=0}^{N} \mathbf{U}_i(t) \phi_i(\mathbf{x}) \tag{6.4-5a}$$

and

$$\hat{p}(\mathbf{x},t) = \sum_{i=0}^{N} p_i(t)\phi_i(\mathbf{x}). \tag{6.4-5b}$$

For convenience, we shall also keep track of the effective stress by adopting a finite-element representation for $\mathbf{t}$:

$$\hat{\mathbf{t}}(\mathbf{x},t) = \sum_{i=0}^{N} \mathbf{t}_i(t)\phi_i(\mathbf{x}). \tag{6.4-5c}$$

In these expansions, $\mathbf{U}_i$, p_i, and $\mathbf{t}_i$ are coefficients to be determined. The reader should be careful not to confuse the unindexed symbol ϕ, signifying the porosity, with the basis functions $\phi_i(\mathbf{x})$.

The substitution of the trial functions (6.4-5) into the governing equations yields a residual, which Galerkin's method forces to be orthogonal to each basis function ϕ_i associated with a non-Dirichlet node. Thus we obtain a set of equations of the form

$$\int_\Omega [\nabla \cdot \hat{\mathbf{t}} - \nabla \cdot (\hat{p}\mathbf{1})]\phi_j(\mathbf{x})\, d\mathbf{x} = \mathbf{0}, \tag{6.4-6a}$$

$$\int_\Omega \left[\frac{K}{\rho g}\nabla^2\hat{p} - \phi\beta\frac{\partial\hat{p}}{\partial t} - \frac{\partial}{\partial t}(\nabla\cdot\hat{\mathbf{U}})\right]\phi_j(\mathbf{x})\, d\mathbf{x} = 0, \tag{6.4-6b}$$

where Ω denotes the spatial domain of the problem. Application of Green's theorem to the first term in Equations (6.4-6a) and (6.4-6b) and introduction of the constitutive relationship expressed in Equation (6.4-4) yields

$$\int_\Omega \left[\lambda(\nabla\cdot\hat{\mathbf{U}})\mathbf{1}\cdot\nabla\phi_j + \mu(\nabla\hat{\mathbf{U}}\cdot\nabla\phi_j + \hat{\mathbf{U}}\nabla\cdot\nabla\phi_j) + (\nabla\hat{p})\phi_j\right] d\mathbf{x} \tag{6.4-7a}$$

$$= \oint_{\partial\Omega} (\mathbf{n}\cdot\hat{\mathbf{t}})\phi_j\, d\mathbf{x}$$

and

$$\int_\Omega \left[\frac{K}{\rho g}\nabla\hat{p}\cdot\nabla\phi_j + \phi\beta\frac{\partial\hat{p}}{\partial t}\phi_j + \frac{\partial}{\partial t}(\nabla\cdot\hat{\mathbf{U}})\phi_j\right] d\mathbf{x} \tag{6.4-7b}$$

$$= \oint_{\partial\Omega} \frac{K}{\rho g}\mathbf{n}\cdot(\phi_j\nabla\hat{p})\, d\mathbf{x}.$$

Consider now the case when Ω is two-dimensional, with x and z being the two Cartesian coordinate directions. Substitution of the definitions of $\hat{\mathbf{U}}$ and $\hat{p}$ from Equations (6.4-5) into the integral equations (6.4-7) yields

three sets of equations. For the first component of Equation (6.4-7a) we get

$$\text{(6.4-8a)} \qquad \sum_{i=0}^{N} \int_{\Omega} \left[\left(\lambda \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + 2\mu \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + \mu \frac{\partial \phi_i}{\partial z} \frac{\partial \phi_j}{\partial z} \right) (U_x)_i \right.$$

$$\left. + \left(\lambda \frac{\partial \phi_i}{\partial z} \frac{\partial \phi_j}{\partial x} + \mu \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial z} \right) (U_z)_i + \frac{\partial \phi_i}{\partial x} \phi_j p_i \right] d\mathbf{x}$$

$$= \oint_{\partial\Omega} (n_x t_{xx} + n_z t_{zx}) \phi_j \, d\mathbf{x},$$

where $(U_x)_i$ and $(U_z)_i$ stand for the x- and z- components of the unknown coefficient $\mathbf{U}_i$. For the second component of Equation (6.4-7a) we get

$$\text{(6.4-8b)} \qquad \sum_{i=0}^{N} \int_{\Omega} \left[\left(\lambda \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial z} + \mu \frac{\partial \phi_i}{\partial z} \frac{\partial \phi_j}{\partial x} \right) (U_x)_i \right.$$

$$\left. + \left(\lambda \frac{\partial \phi_i}{\partial z} \frac{\partial \phi_j}{\partial z} + 2\mu \frac{\partial \phi_i}{\partial z} \frac{\partial \phi_j}{\partial z} + \mu \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} \right) (U_z)_i + \frac{\partial \phi_i}{\partial z} \phi_j p_i \right] d\mathbf{x}$$

$$= \oint_{\partial\Omega} (n_x t_{xz} + n_z t_{zz}) \phi_j \, d\mathbf{x}.$$

Finally, for Equation (6.4-7b), we get

$$\text{(6.4-8c)} \qquad \sum_{i=0}^{N} \int_{\Omega} \left[\frac{\partial \phi_i}{\partial x} \phi_j \frac{d(U_x)_i}{dt} + \frac{\partial \phi_i}{\partial z} \phi_j \frac{d(U_z)_i}{dt} \right.$$

$$\left. + \frac{K}{\rho g} \left(\frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial z} \frac{\partial \phi_j}{\partial z} \right) p_i + \phi \beta \phi_i \phi_j \frac{dp_i}{dt} \right] d\mathbf{x}$$

$$= \oint_{\partial\Omega} \frac{K}{\rho g} \left(n_x \frac{\partial p}{\partial x} + n_z \frac{\partial p}{\partial z} \right) \phi_j \, d\mathbf{x}.$$

The set of equations (6.4-8) can be conveniently written in matrix form. The equations for the unknown variables at a single spatial node i are as follows:

$$\text{(6.4-9)} \qquad \begin{bmatrix} A_{ji} & B_{ji} & C_{ji} \\ D_{ji} & E_{ji} & G_{ji} \\ 0 & 0 & L_{ji} \end{bmatrix} \begin{bmatrix} (U_x)_i \\ (U_z)_i \\ p_i \end{bmatrix}$$

$$+\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ H_{ji} & K_{ji} & M_{ji} \end{bmatrix} \frac{d}{dt} \begin{bmatrix} (U_x)_i \\ (U_z)_i \\ p_i \end{bmatrix} = \begin{bmatrix} (F_x)_j \\ (F_z)_j \\ (F_p)_j \end{bmatrix}.$$

Note that this matrix equation constitutes one block in the global matrix equation for the unknown coefficients at all nodes. The functional forms of the matrix entries in Equation (6.4-9) can be determined through inspection of Equations (6.4-8); they are as follows:

$$\begin{aligned}
A_{ji} &= \int_\Omega \left[(\lambda + 2\mu) \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + \mu \frac{\partial \phi_i}{\partial z} \frac{\partial \phi_j}{\partial z} \right] d\mathbf{x}, \\
B_{ji} &= \int_\Omega \left(\lambda \frac{\partial \phi_i}{\partial z} \frac{\partial \phi_j}{\partial x} + \mu \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial z} \right) d\mathbf{x}, \\
C_{ji} &= \int_\Omega \frac{\partial \phi_i}{\partial x} \phi_j \, d\mathbf{x} = H_{ji}, \\
D_{ji} &= \int_\Omega \left(\lambda \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial z} + \mu \frac{\partial \phi_i}{\partial z} \frac{\partial \phi_j}{\partial x} \right) d\mathbf{x}, \\
E_{ji} &= \int_\Omega \left[(\lambda + 2\mu) \frac{\partial \phi_i}{\partial z} \frac{\partial \phi_j}{\partial z} + \mu \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} \right] d\mathbf{x}, \\
G_{ji} &= \int_\Omega \frac{\partial \phi_i}{\partial z} \phi_j \, d\mathbf{x} = K_{ji}, \\
L_{ji} &= \int_\Omega \frac{K}{\rho g} \left(\frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial z} \frac{\partial \phi_j}{\partial z} \right) d\mathbf{x}, \\
M_{ji} &= \int_\Omega \phi \beta \phi_i \phi_j \, d\mathbf{x}.
\end{aligned}$$

The entries of the forcing vector are just integrals of the known boundary values:

$$\begin{aligned}
(F_x)_j &= \oint_{\partial\Omega} (n_x t_{xx} + n_z t_{zx}) \phi_j \, d\mathbf{x}, \\
(F_z)_j &= \oint_{\partial\Omega} (n_x t_{xz} + n_z t_{zz}) \phi_j \, d\mathbf{x}, \\
(F_p)_j &= \oint_{\partial\Omega} \frac{K}{\rho g} \left(n_x \frac{\partial p}{\partial x} + n_z \frac{\partial p}{\partial z} \right) \phi_j \, d\mathbf{x}.
\end{aligned}$$

We can reduce the set of ordinary differential equations (6.4-9) to a set of algebraic equations using a standard finite-difference approximation

of the time derivatives. Thus, using a time step of length k, we get

$$\left.\frac{d(U_x)_i}{dt}\right|^{n+\theta} = \frac{1}{k}[(U_x)_i^{n+1} - (U_x)_i^n] + \mathcal{O}(k),$$

$$\left.\frac{d(U_z)_i}{dt}\right|^{n+\theta} = \frac{1}{k}[(U_z)_i^{n+1} - (U_z)_i^n] + \mathcal{O}(k),$$

$$\left.\frac{dp_i}{dt}\right|^{n+\theta} = \frac{1}{k}(p_i^{n+1} - p_i^n) + \mathcal{O}(k).$$

(The truncation errors for the case $\theta = 1/2$ will be $\mathcal{O}(k^2)$.) Corresponding to these difference approximations we have the following approximations of the unknown nodal values in time:

$$(U_x)_i^{n+\theta} \equiv \theta(U_x)_i^{n+1} + (1-\theta)(U_x)_i^n,$$
$$(U_z)_i^{n+\theta} \equiv \theta(U_z)_i^{n+1} + (1-\theta)(U_z)_i^n,$$
$$p_i^{n+\theta} \equiv \theta p_i^{n+1} + (1-\theta)p_i^n.$$

With this temporal discretization, Equations (6.4-9) can be rewritten in the matrix form

$$\text{(6.4-10)} \quad \left(\theta\begin{bmatrix} A_{ji} & B_{ji} & C_{ji} \\ D_{ji} & E_{ji} & G_{ji} \\ 0 & 0 & L_{ji} \end{bmatrix} + \frac{1}{k}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ H_{ji} & K_{ji} & M_{ji} \end{bmatrix}\right)\begin{bmatrix} (U_x)_i^{n+1} \\ (U_z)_i^{n+1} \\ p_i^{n+1} \end{bmatrix}$$

$$+\left((1-\theta)\begin{bmatrix} A_{ji} & B_{ji} & C_{ji} \\ D_{ji} & E_{ji} & G_{ji} \\ 0 & 0 & L_{ji} \end{bmatrix} - \frac{1}{k}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ H_{ji} & K_{ji} & M_{ji} \end{bmatrix}\right)\begin{bmatrix} (U_x)_i^n \\ (U_z)_i^n \\ p_i^n \end{bmatrix}$$

$$= \begin{bmatrix} (F_x)_j^{n+\theta} \\ (F_z)_j^{n+\theta} \\ (F_p)_j^{n+\theta} \end{bmatrix}.$$

Given initial and boundary conditions, one can solve the global matrix equations generated by the nodal equations (6.4-10) by proceeding stepwise through time. At the end of each time step, the finite-element nodes are moved according to the calculated displacements $\mathbf{U}(\mathbf{x}, t)$, and the new geometry of the region is then used to compute the next time step. For a more detailed explanation of this procedure, we refer the reader to Safai (1977).

6.5. Oil Reservoir Modeling.

One of the most active areas of current research and development in large-scale numerical modeling is the field of oil reservoir simulation. An oil reservoir is a deposit of petroleum and associated fluids, usually gas and brine, held underground in the interstices of a porous rock formation. To produce the oil, petroleum engineers must use the natural forces in the reservoir, such as pressure, together with artificial techniques such as fluid injection to overcome the actions of viscosity and capillary forces that impede the flow of oil into production wells. In devising optimal production strategies, reservoir engineers commonly use mathematical simulators to assess the effects of various fluid production and injection schemes on the flow of oil, gas, and water in the rock formation

Our purpose in this section is twofold. First, we wish to show how the basic equations of petroleum reservoir flow arise from the continuum-mechanical considerations presented in Chapter One. Second, we wish to outline some of the most common numerical schemes used to approximate these governing equations. Readers interested in more thorough introductions to oil reservoir simulation should consult monographs by Peaceman (1977) and Aziz and Settari (1979) together with the other references cited below.

Compositional flows in porous media.

Oil reservoirs are mixtures. They consist of several phases, including rock, oil, water, and gas, and therefore the theory of multiphase mixtures applies. They also consist of many molecular species. The hydrocarbon species in the oil and gas phases are especially important, since transfers of species between the oil and gas phases, including the processes of dissolution, evaporation, condensation, and gas percolation, often have profound effects on a reservoir's production. Thus the theory of multispecies mixtures is important here, too. Flows involving multiphase, multispecies mixtures are called **compositional** flows. To accommodate the presence of both phases and species, we shall begin by extending the mixture-theoretic formalism outlined in Section 1.5.

For simplicity, let us assume that there are three fluid phases in the reservoir, namely, water (W), oil (O), and gas (G), with chemical species indexed by $i = 1, \ldots, N+1$. Let us label the rock phase by the index R. Conceivably, at least, each species can exist in any phase and can transfer between phases via dissolution, evaporation, condensation, and so forth, subject to thermodynamic constraints. We shall assume here that the rock is chemically inert and that there are no intraphase or stoichiometric chemical reactions, although in some applications to certain enhanced oil recovery technologies reactions of this kind may be important.

In this mixture, each pair (i, α), with i chosen from the species indices and α chosen from the phases, is a constituent. Thus, for example, CH_4 in the gas phase is one constituent, CH_4 in oil another, and n-C_4H_{10} in oil yet another. Each constituent (i, α) has its own **intrinsic mass density** ρ_i^α, measured as mass of i per unit volume of α, and its own velocity $\mathbf{v}_i^\alpha$. To accommodate the established kinematics of phases, we shall still associate with each phase α its volume fraction ϕ_α. Moreover, letting $\phi = 1 - \phi_R$ be the porosity, we define the **saturation** of fluid phase α as $S_\alpha = \phi_\alpha/\phi$. Using these basic quantities, we then define several useful variables. The **intrinsic mass density of phase** α is

$$\rho^\alpha = \sum_{i=1}^{N} \rho_i^\alpha;$$

the **mass fraction of species** i **in phase** α is

$$\omega_i^\alpha = \rho_i^\alpha/\rho^\alpha;$$

the **bulk density of the fluids** is

$$\rho = \phi \sum_{\alpha \neq R} S_\alpha \rho^\alpha;$$

the **total mass fraction of species** i **in the fluids** is

$$\omega_i = (\phi/\rho) \sum_{\alpha \neq R} S_\alpha \rho^\alpha \omega_i^\alpha;$$

the **barycentric velocity of phase** α is

$$\mathbf{v}^\alpha = (1/\rho^\alpha) \sum_{i=1}^{N} \rho_i^\alpha \mathbf{v}_i^\alpha;$$

and the **diffusion velocity of species** i **in phase** α is

$$\mathbf{u}_i^\alpha = \mathbf{v}_i^\alpha - \mathbf{v}^\alpha.$$

Suppose the index $N + 1$ represents the species making up the inert rock phase. Then the following constraints hold:

$$\sum_{i=1}^{N} \omega_i = \sum_{i=1}^{N} \omega_i^\alpha = \sum_{\alpha} \phi_\alpha = \sum_{\alpha \neq R} S_\alpha = 1,$$

where the index α in the second sum can represent any fluid phase, and

$$\sum_{i=1}^{N} \rho_i^\alpha \mathbf{u}_i^\alpha = 0.$$

Each constituent (i, α) has its own mass balance, given by analogy with Equation (1.5-2) as

$$\frac{\partial}{\partial t}(\phi_\alpha \rho_i^\alpha) + \nabla \cdot (\phi_\alpha \rho_i^\alpha \mathbf{v}_i^\alpha) = r_i^\alpha,$$

where the exchange terms r_i^α must obey the restriction $\sum_{i=1}^{N} \sum_{\alpha \neq R} r_i^\alpha = 0$. If we impose the further constraint that there are no intraphase chemical reactions, then we have in addition $\sum_{\alpha \neq R} r_i^\alpha = 0$ for each species $i = 1, \ldots, N$. Since phase velocities are typically more accessible to measurement than species velocities, it is convenient to rewrite the constituent mass balance, following the species balance derivation in Section 1.5, as

$$\frac{\partial}{\partial t}(\phi S_\alpha \rho^\alpha \omega_i^\alpha) + \nabla \cdot (\phi S_\alpha \rho^\alpha \omega_i^\alpha \mathbf{v}^\alpha) + \nabla \cdot \mathbf{j}_i^\alpha = r_i^\alpha,$$

where $\mathbf{j}_i^\alpha = \phi S_\alpha \rho^\alpha \omega_i^\alpha \mathbf{u}_i^\alpha$ stands for the **diffusive flux** of constituent (i, α). Summing this equation over all fluid phases α and using the restrictions gives a total mass balance for each species i:

$$\frac{\partial}{\partial t}(\rho \omega_i) + \nabla \cdot [\phi(S_W \rho^W \omega_i^W \mathbf{v}^W + S_O \rho^O \omega_i^O \mathbf{v}^O + S_G \rho^G \omega_i^G \mathbf{v}^G)]$$

$$+ \nabla \cdot (\mathbf{j}_i^W + \mathbf{j}_i^O + \mathbf{j}_i^G) = 0, \qquad i = 1, \ldots, N.$$

To establish flow equations for each species, we need velocity field equations for each fluid phase together with some constitutive equations for the diffusive fluxes $\mathbf{j}_i^\alpha$. For the fluid velocities we may postulate Darcy's law, Equation (1.5-6), assuming that the porous medium is isotropic. However, when several fluid phases occupy the porous rock, we must alter Darcy's law to account for the interference to flow that each fluid feels when other fluids occupy the same pore space. The most common way to do this is to write, for each fluid α,

$$\mathbf{v}^\alpha = -\frac{k k_{r\alpha}}{\mu^\alpha \phi S_\alpha}(\nabla p^\alpha - \rho^\alpha g \nabla Z).$$

The factor $k_{r\alpha}$ is called the **relative permeability** of fluid α and satisfies $0 \leq k_{r\alpha} \leq 1$. In many cases it is reasonable to assume that $k_{r\alpha}$ is a function

of fluid saturations only for a given rock-fluid system. Figure 6-5 shows a typical pair of relative permeability functions for an oil-water-rock mixture.

The existence of several fluid phases also allows for the existence of several fluid pressures. This effect actually occurs in nature, owing to the physics of fluid-fluid and fluid-rock interfaces. The difference between two fluid pressures at a given point in the reservoir is the **capillary pressure**, defined by $p_{C\alpha\beta} = p_\alpha - p_\beta$. Clearly, when three fluid phases exist, only two capillary pressures can be independent. Capillary pressures, like relative permeabilities, are typically functions of fluid saturation in a given rock-fluid system.

For the diffusive fluxes the appropriate assumption is not so clear. In single-phase flows through porous media, the diffusive flux of a species with respect to the fluid's barycentric velocity is called **hydrodynamic dispersion.** The literature on single-phase hydrodynamic dispersion is large but by no means conclusive. By contrast, the literature on hydrodynamic dispersion in multifluid flows is quite sparse. The most common approach in multiphase oil reservoir simulation is to assume that hydrodynamic dispersion is a small enough effect that the diffusive fluxes in the mass balance for each species are negligible. Provided this assumption is reasonable, we arrive at the following flow equation for species i in the fluids:

$$\text{(6.5-1)} \qquad \frac{\partial}{\partial t}[\phi(S_W \rho^W \omega_i^W + S_O \rho^O \omega_i^O + S_G \rho^G \omega_i^G)]$$

$$-\nabla \cdot \left[\frac{k k_{rW} \rho^W \omega_i^W}{\mu_W} (\nabla p_W - \rho^W g \nabla Z) + \frac{k k_{rO} \rho^O \omega_i^O}{\mu_O} (\nabla p_O - \rho^O g \nabla Z) \right.$$

$$\left. + \frac{k k_{rG} \rho^G \omega_i^G}{\mu_G} (\nabla p_G - \rho^G g \nabla Z) \right] = 0, \qquad i = 1, \ldots, N.$$

To close this set of equations, we need some supplementary constraints giving relationships among the variables. One class of supplementary constraints consists of the thermodynamic relationships giving phase densities and compositions as functions of pressure and overall fluid mixture composition. Conceptually, these relationships take the forms

$$\begin{aligned}
\rho^\alpha &= \rho^\alpha(\omega_1^\alpha, \ldots, \omega_{N-1}^\alpha, p_\alpha), && \alpha = W, O, G, \\
\omega_i^\alpha &= \omega_i^\alpha(\omega_1, \ldots, \omega_{N-1}, p_\alpha), && \alpha = W, O, G; \quad i = 1, \ldots, N-1, \\
S_\alpha &= S_\alpha(\omega_1, \ldots, \omega_{N-1}, p_\alpha), && \alpha = W, O, G.
\end{aligned}$$

However, it is important from a computational viewpoint to observe that the actual thermodynamic statements of these relationships may yield simultaneous sets of nonlinear algebraic equations giving phase densities,

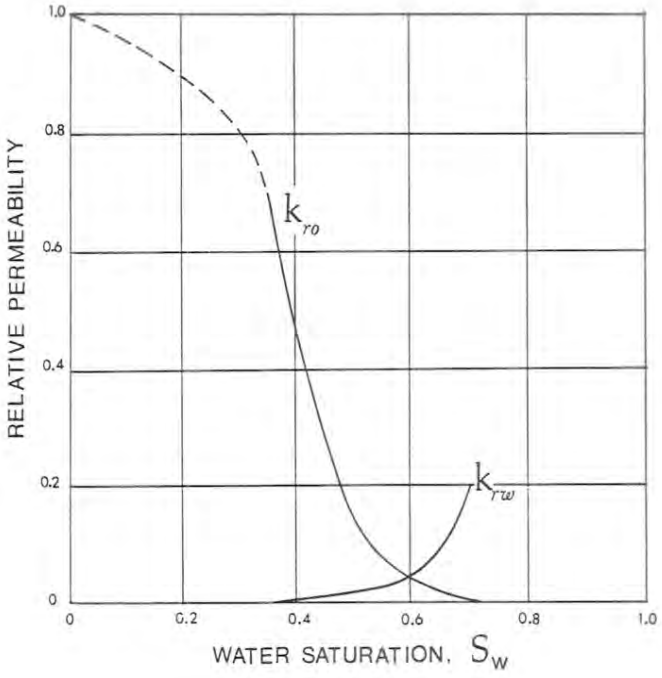

Figure 6-5. Typical relative permeability curves for water-wet rock matrix.

compositions, and saturations implicitly. If this is the case, then the calculation of fluid-phase thermodynamics may constitute a major part of the computational effort in a simulation. We refer the interested reader to Coats (1980) or Allen (1984a) for discussions of two different approaches to modeling flows with such complicated thermodynamics.

The other class of supplementary constraints includes constitutive relationships for the particular rock-fluid system being modeled. These relationships involve the capillary pressures and relative permeabilities, typically taking the forms

$$\begin{aligned} p_O - p_W &= p_{COW} = p_{COW}(S_O, S_G), \\ p_G - p_O &= p_{CGO} = p_{CGO}(S_O, S_G), \\ k_{r\alpha} &= k_{r\alpha}(S_O, S_G), \qquad \alpha = W, O, G. \end{aligned}$$

In practice, petroleum engineers derive these functions through measurements of fluid and rock samples extracted from the reservoir under study.

Black-oil models.

Black-oil models are special cases of the general compositional equations that allow limited interphase mass transfer, the composition of each phase depending on pressures only. This class of models has become a standard engineering tool in the petroleum industry. We shall review the formulation of the black-oil equations and discuss a few selected aspects of their numerical solution.

The fundamental premise of the black-oil model is that a highly simplified, three-species system can often serve as an adequate model of the complex mixtures of brine and hydrocarbons found in natural petroleum reservoirs. For practical purposes, petroleum engineers define these three "pseudo-species" according to what appears at the surface, at **stock-tank conditions** (STC), after production of the reservoir fluids. Thus, we have the species o, which is stock-tank oil; g, which is stock-tank gas, and w, which is stock-tank water. Underground, at **reservoir conditions** (RC), these species may partition themselves among the three fluid phases O, G, and W in a distribution depending on the pressures in the formation.

Now we impose a set of thermodynamic constraints on this partitioning of species. First, we assume that there is no exchange of water w into the nonaqueous phases O and G, so that $\omega_w^W = 1$, and $\omega_w^O = \omega_w^G = 0$. Second, we allow no exchange of oil o into the vapor phase G or the aqueous liquid W, so that $\omega_o^O = 1$, and $\omega_o^W = \omega_o^G = 0$. Third, we prohibit the dissolution of gas g into the aqueous liquid W, so that $\omega_g^W = 0$. However, we allow the gas g to dissolve in the hydrocarbon liquid O according to a pressure-dependent relationship called the **solution gas-oil ratio**, defined by

$$R_S(p_O) = \frac{\text{volume of } g \text{ in solution at RC}}{\text{volume of } o},$$

where the volumes refer to volumes at STC.

To facilitate further reference to volumes of species at STC, we relate the phase densities ρ^α at RC to the species densities ρ_i^{STC} at STC by defining the **formation volume factors**. For W and G these definitions are fairly simple:

$$B_W(p_W) = \frac{\rho_w^{\mathrm{STC}}}{\rho^W(p_W)}, \qquad B_G(p_W) = \frac{\rho_g^{\mathrm{STC}}}{\rho^G(p_G)}.$$

For the hydrocarbon liquid O, however, we must also account for the mass of dissolved gas at RC:

$$B_O(p_O) = \frac{\rho_o^{\mathrm{STC}} + R_S(p_O)\rho_g^{\mathrm{STC}}}{\rho^O(p_O)}.$$

If we substitute these definitions into the flow equations (6.5-1) for the species o, g, w and divide through by the constants ρ_i^{STC}, we obtain the three **black-oil equations**. The flow equations for water, oil, and gas are, respectively,

$$\frac{\partial}{\partial t}\left(\frac{\phi S_W}{B_W}\right) - \nabla\cdot[\lambda_W(\nabla p_W - \gamma_W \nabla Z)] = 0; \tag{6.5-2a}$$

$$\frac{\partial}{\partial t}\left(\frac{\phi S_O}{B_O}\right) - \nabla\cdot[\lambda_O(\nabla p_O - \gamma_O \nabla Z)] = 0, \tag{6.5-2b}$$

$$\frac{\partial}{\partial t}\left[\phi\left(\frac{S_G}{B_G} + \frac{R_S S_O}{B_O}\right)\right] - \nabla\cdot[\lambda_G(\nabla p_G - \gamma_G \nabla Z)] - \nabla\cdot[R_S\lambda_O(\nabla p_O - \gamma_O \nabla Z)] = 0. \tag{6.5-2c}$$

To keep the notation tractable, we have adopted the abbreviations $\lambda_\alpha = kk_{r\alpha}/(\mu_\alpha B_\alpha)$ and $\gamma_\alpha = \rho^\alpha g$.

These equations constitute a system of coupled, nonlinear, transient PDEs. As we have argued throughout this book, the numerical solution methodology appropriate for a given system depends to a great extent on the classification of the governing PDEs. Although each of the black-oil equations is formally parabolic in appearance, the system can often exhibit behavior more typical of an elliptic-hyperbolic set if capillary influences are small. To see this, consider the two-phase version of Equations (6.5-2) in which gas is absent, porosity is constant, and fluid compressibilities and gravity forces have no effect. The flow equations in this simple case reduce to

$$-\phi\frac{\partial S_W}{\partial t} = \nabla\cdot(\lambda_O \nabla p_O),$$

$$\phi\frac{\partial S_W}{\partial t} = \nabla\cdot(\lambda_W \nabla p_W),$$

since $S_O + S_W = 1$. Adding these two equations gives a total flow equation $\nabla \cdot \mathbf{Q} = 0$, where $\mathbf{Q} = -\lambda_O \nabla p_O - \lambda_W \nabla p_W$ represents the total rate of fluid flow. Calling $\lambda = \lambda_O + \lambda_W$ and $p = (p_O + p_W)/2$, we can rewrite this total flow equation as follows:

$$\nabla \cdot \left[\lambda \nabla p - \left(\frac{\lambda_W - \lambda_O}{2} \right) \nabla p_{COW} \right] = 0.$$

If we examine the case when $\nabla p_{COW} \simeq \mathbf{0}$, the total flow equation reduces to an elliptic **pressure equation**,

$$\nabla \cdot (\lambda \nabla p) = 0.$$

We discussed the solution of such equations in depth in Chapter Three. Next, defining the **fractional flow function** $f_W = \lambda_W/(\lambda_O + \lambda_W)$, we can rewrite the water flow equation as a hyperbolic PDE,

$$\phi \frac{\partial S_W}{\partial t} + \nabla \cdot [\mathbf{Q} f_W(S_W)] = 0.$$

This is the three-dimensional **Buckley-Leverett equation** (Buckley and Leverett, 1942). We discussed the solution of such nonlinear hyperbolic conservation laws in some detail in Chapter Five. The Buckley-Leverett equation demonstrates that, in an oil reservoir where capillary effects are small, fluid saturations can flow according to essentially hyperbolic PDEs in response to an elliptic pressure field.

Several approaches to solving the general system (6.5-2) numerically have appeared in the petroleum engineering literature. Much of the classic work on black-oil models has focused on schemes for advancing the system of coupled PDEs in time. We shall review two of the most popular methods: the **simultaneous solution** (SS) method and the **implicit pressure–explicit saturation** (IMPES) method.

Simultaneous solution (SS).

The SS method, introduced by Douglas, Peaceman, and Rachford (1959) and further developed by Coats et al. (1967), treats the flow equations (6.5-2) as simultaneous equations for the fluid pressures p_O, p_G, and p_W. Inverting the capillarity relationships and imposing the restriction on fluid saturations then yields the saturations S_O, S_G, and S_W. For ease of presentation, let us examine the two-phase case, assuming that the vapor phase G does not appear and that the porosity ϕ is constant.

The first step in the formulation is to rewrite the flow equations so that the pressures p_O and p_W appear as explicit unknowns. To do this, we apply the chain rule to the accumulation terms, giving

$$\frac{\partial}{\partial t}\left(\frac{\phi S_W}{B_W} \right) = \phi S_W b_W \frac{\partial p_W}{\partial t} + \frac{\phi S'_W}{B_W} \left(\frac{\partial p_O}{\partial t} - \frac{\partial p_W}{\partial t} \right),$$

$$\frac{\partial}{\partial t}\left(\frac{\phi S_O}{B_O}\right) = \phi S_O b_O \frac{\partial p_O}{\partial t} - \frac{\phi S'_W}{B_O}\left(\frac{\partial p_O}{\partial t} - \frac{\partial p_W}{\partial t}\right),$$

where $b_\alpha = d(1/B_\alpha)/dp_\alpha$ and S'_W signifies the derivative of the inverted capillarity relationship $S_W(p_{COW})$. This device allows us to write the system (6.5-2) as follows:

$$\phi \begin{bmatrix} (S_W b_W - S'_W/B_W) & (S'_W/B_O) \\ (S'_W/B_O) & (S_O b_O - S'_W/B_O) \end{bmatrix} \frac{\partial}{\partial t}\begin{bmatrix} p_W \\ p_O \end{bmatrix}$$
$$-\nabla \cdot \begin{bmatrix} (\Lambda_W/B_W)\nabla & 0 \\ 0 & (\Lambda_O/B_O)\nabla \end{bmatrix}\begin{bmatrix} p_W \\ p_O \end{bmatrix}$$
$$+\begin{bmatrix} \rho^W g \nabla Z \\ \rho^O g \nabla Z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The terms involving time derivatives in this equation represent the accumulation of fluid, while the terms involving the pressure gradients govern the Darcy flux of fluids.

Now we can employ some finite-difference or finite-element method to approximate the spatial derivative, getting a system of evolution equations having the matrix form

$$\mathbf{M}\frac{d\mathbf{p}}{dt} + \mathbf{S}\mathbf{p} = \mathbf{f}.$$

Here $\mathbf{M}$ is the matrix of coefficients arising from the spatial discretization of the accumulation terms, $\mathbf{S}$ is the matrix of coefficients arising from the discretization of fluid fluxes, $\mathbf{p}$ represents the vector of unknown nodal values of oil and water pressure, and $\mathbf{f}$ is a vector containing information from the discretized boundary conditions. Since the entries of $\mathbf{M}$ and $\mathbf{S}$ exhibit pronounced functional dependence on the unknown pressures, this system is strongly nonlinear. Therefore, as discussed in Section 6.3, our time-stepping approximation must be iterative. As an example, given a time step k_t, we might use the Newton-like procedure introduced at the end of Section 6.3 to advance from $t = nk_t$ to $t = (n+1)k_t$, yielding

$$\left(\frac{1}{k_t}\mathbf{M}^{n+1,m} + \mathbf{S}^{n+1,m}\right)\delta\mathbf{p}^{n+1,m+1}$$
$$= -\left[\frac{1}{k_t}\mathbf{M}^{n+1,m}\left(\mathbf{p}^{n+1,m} - \mathbf{p}^n\right) - \mathbf{S}^{n+1,m}\mathbf{p}^{n+1,m} + \mathbf{f}^{n+1,m}\right]$$
$$= -\mathbf{R}^{n+1,m}.$$

Here, $\delta\mathbf{p}^{n+1,m+1}$ is a vector of iterative increments, so that, starting from iteration level m, $\mathbf{p}^{n+1,m+1} = \mathbf{p}^{n+1,m} + \delta\mathbf{p}^{n+1,m+1}$ gives the vector of nodal pressures at the next iteration for the unknown time level $n+1$. In

this scheme the notation $\mathbf{R}^{n+1,m}$ suggests that we regard the right side as a residual, iterating at each time step until $\|\mathbf{R}^{n+1,m}\|$ is small enough in some norm.

The formulation presented above is not unique. In fact, several variants of the SS method have appeared, including formulations treating different sets of variables as principal unknowns. Aziz and Settari (1979) provide a survey of these alternative approaches.

Implicit pressure-explicit saturation (IMPES).

In the IMPES formulation, the basic idea is to combine the flow equations (6.5-2) to get an equation for one of the fluid pressures (Breitenbach, Thurnau, and van Poolen (1969)). Solving this equation implicitly provides the information necessary to update the saturations explicitly at each time step, using an independent set of flow equations and the restriction that saturations sum to unity. Sheldon, Zondek, and Cardwell (1959) and Stone and Garder (1961) introduced this method.

We begin, as in the SS method, by expanding the accumulation terms, this time leaving saturations and pressures as principal unknowns. For the three-phase system, this leads to the following finite-difference approximations to the time derivatives:

$$\begin{aligned}
\phi\frac{\partial}{\partial t}\left(\frac{S_W}{B_W}\right) &= \frac{1}{k_t}(C_1\Delta_t S_W + C_2\Delta_t p_w) + \mathcal{O}(k_t),\\
\phi\frac{\partial}{\partial t}\left(\frac{S_O}{B_O}\right) &= \frac{1}{k_t}(C_3\Delta_t S_O + C_4\Delta_t p_O) + \mathcal{O}(k_t),\\
\phi\frac{\partial}{\partial t}\left(\frac{S_G}{B_G} + \frac{R_S S_O}{B_O}\right) &= \frac{1}{k_t}(C_5\Delta_t S_G + C_6\Delta_t p_G\\
&\quad + C_7\Delta_t S_O + C_8\Delta_t p_O) + \mathcal{O}(k_t).
\end{aligned}$$

The coefficients $C_1,\ldots,C_8$ appearing here stand for the appropriate derivatives extracted using the chain rule. The reader should derive expressions for these coefficients. The notation $\Delta_t u = u^{n+1} - u^n$ defines the time-difference operator.

The next step involves the crucial assumption that the capillary pressures p_{COW} and p_{CGO} change negligibly over a time step. This assumption implies that $\Delta_t p_O = \Delta_t p_W = \Delta_t p_G$ and, furthermore, that we can treat the capillary contributions to the flux terms explicitly. Thus, our implicit, temporally discrete approximations to Equations (6.5-2) become

$$\begin{aligned}
&\text{(6.5-3a)}\quad C_1\Delta_t S_W + C_2\Delta_t p_O\\
&\qquad = k_t\nabla\cdot[\lambda_W^{n+1}(\nabla p_O^{n+1} - \nabla p_{COW}^{n} - \gamma_W^{n+1}\nabla Z)],\\
&\text{(6.5-3b)}\quad C_3\Delta_t S_O + C_4\Delta_t p_O = k_t\nabla\cdot[\lambda_O^{n+1}(\nabla p_O^{n+1} - \gamma_O^{n+1}\nabla Z)],
\end{aligned}$$

$$
\begin{aligned}
(6.5\text{-}3c)\quad & C_5\Delta_t S_G + C_7\Delta_t S_O + (C_6 + C_8)\Delta_t p_O \\
& = k_t\nabla\cdot[\lambda_G^{n+1}(\nabla p_O^{n+1} + \nabla p_{CGO}^{n+1} - \gamma_G^{n+1}\nabla Z) \\
& + R_S^{n+1}\lambda_O^{n+1}(\nabla p_O^{n+1} - \gamma_O^{n+1}\nabla Z)].
\end{aligned}
$$

To get a single pressure equation from this set, we multiply Equation (6.5-3c) by the coefficient $B = C_3/(C_5 - C_7)$ and multiply Equation (6.5-3a) by $A = BC_5/C_1$. Now add Equations (6.5-3a–c). Observe that the saturation differences in the accumulation terms arising from Equations (6.5-3) in this way now sum to an expression proportional to $\Delta_t(S_W + S_O + S_G) = 0$. Therefore our weighted sum of the time-differenced flow equations yields

$$
\begin{aligned}
(6.5\text{-}4)\quad C^{n+1}\Delta_t p_O = k_t\{ & A^{n+1}\nabla\cdot\left[\lambda_W^{n+1}\left(\nabla p_O^{n+1} - \nabla p_{COW}^{n}\right)\right] \\
& + \nabla\cdot(\lambda_O^{n+1}\nabla p_O^{n+1}) \\
& + B^{n+1}\nabla\cdot\left[(\lambda_G^{n+1} + R_S^{n+1}\lambda_O^{n+1})\nabla p_O^{n+1}\right. \\
& \left. + \lambda_G^{n+1}\nabla p_{CGO}^{n}\right] - \Gamma^{n+1}\}.
\end{aligned}
$$

The new parameter Γ is shorthand for the weighted sum of the gravity terms, and $C = AC_2 + C_4 + B(C_6 + C_8)$. Equation (6.5-4) is the desired pressure equation.

Now, provided we have an appropriate technique for producing discrete approximations to the spatial derivatives appearing in these equations, we can implement the following time-stepping procedure.

(i) Solve the nonlinear pressure equation (6.5-4) implicitly, using some iterative scheme such as those introduced in Section 6.3.

(ii) Solve Equation (6.5-3a) explicitly for $\Delta_t S_W$ and update the water saturation; solve (6.5-3b) for $\Delta_t S_O$ and update the oil saturation; then set $S_G^{n+1} = 1 - S_W^{n+1} - S_O^{n+1}$.

(iii) Compute p_{COW}^{n+1} and p_{CGO}^{n+1} using the new saturations; then use these to update p_W and p_G.

(iv) Begin the next time step.

As with the SS methods, variants on this development have appeared; see Aziz and Settari (1979) for a survey.

Notice that, in contrast to the SS formulation, the IMPES approach requires the implicit solution of only one flow equation at each time step. Thus IMPES schemes offer the obvious advantage that, with only one implicit equation to solve per time step, the algorithm requires smaller matrix inversions at each iteration. The resulting computational savings can be significant in problems involving large numbers of grid points. On the other hand, because it treats capillary pressures explicitly, the IMPES method

suffers from instability when the time step k_t exceeds a critical value. This limitation can be inconvenient if the critical value of k_t is unknown or small compared with the life of a field project. The SS method, while requiring more computation per time step, boasts greater stability. This can prove to be a decided advantage when the problem to be solved exhibits strongly nonlinear phenomena such as gas percolation, when the pressures of liquid hydrocarbons pass through bubble points and enter the gas phase.

The temporal weighting of the flux coefficients λ_α also affects the stability of discrete solutions to the black-oil equations. It is a fairly common practice to treat these coefficients explicitly. However, this tactic leads to limits on time steps allowable for stable solutions. The limitation is especially severe in problems with gas percolation. The implicit treatment of the flux coefficients partially alleviates this stability problem.

Matrix computations.

One of the most important problems in black-oil simulation is the computational inefficiency associated with the solution of large systems of linear algebraic equations. In either the SS or the IMPES approach, the iterative time-stepping scheme calls for the solution of matrix equations at each iteration of each time step. For simulations at practical scales these calculations alone can tax the storage and CPU-time resources of the largest machines currently available. A great deal of research has focused on the development of fast iterative techniques for the solution of the large matrix systems arising in applications. Early investigations along this line explored the use of various forms of relaxation, as introduced in Section 3.13, to solve the matrix equations. The block-iterative techniques, also discussed in Section 3.13, have been especially attractive in this regard. Alternating-direction iterative techniques, as reviewed in Section 4.3, have also attracted interest. More recently, algorithms of the preconditioned conjugate-gradient type, as outlined in Section 3.15, have proven to be effective.

Some considerations for spatial discretization.

So far we have left hanging the discretization of the spatial derivatives that appear in the black-oil equations. Historically, the method of finite differences has dominated the petroleum engineering literature. The approximation of spatial derivatives using difference analogs in Equations (6.5-2) is relatively straightforward except for one special consideration: Most black-oil simulators employ upstream-weighted approximations to the flux coefficients λ_α. In the simple case of one-dimensional flow, for example, the discretization of a term like $\partial(\lambda\partial p/\partial x)/\partial x)$ leads to a difference

analog of the form

$$\frac{1}{h}\left[\lambda_{i+\frac{1}{2}}\left(\frac{p_{i+1}-p_i}{h}\right)-\lambda_{i-\frac{1}{2}}\left(\frac{p_i-p_{i-1}}{h}\right)\right], \tag{6.5-5}$$

where h is the grid spacing. In its most primitive form, upstream weighting calls for evaluating $\lambda_{i\pm\frac{1}{2}}$ at integer node numbers $i-1$, i, or $i+1$ as follows:

$$\lambda_{i\pm\frac{1}{2}}=\begin{cases}\lambda_{i\pm\frac{1}{2}-\frac{1}{2}}, & \text{if flow is from } i-1 \text{ to } i+1,\\ \lambda_{i\pm\frac{1}{2}+\frac{1}{2}}, & \text{if flow is from } i+1 \text{ to } i-1.\end{cases}$$

The motivation for upstream weighting arises from the partly hyperbolic character of the flow equations, as discussed previously. In Chapter Five we mentioned the lack of uniqueness associated with nonlinear hyperbolic conservation laws and the need to guarantee that solutions to such equations reflect the limiting behavior of their dissipative counterparts. When capillary effects are globally small compared with pressure forces, high-order spatial discretizations can fail to capture the physically important dissipation and therefore converge to incorrect solutions. Upstream weighting introduces an $\mathcal{O}(h)$ error term that restores the necessary dissipation in a numerically consistent fashion. We refer the reader to Allen (1984b) for a more detailed analysis.

The reader should beware that, in highly complex systems of PDEs, upstream weighting has its own perils. In particular, simple extensions of the scheme given in Equation (6.5-5) to several spatial variables can yield numerical schemes that suffer from physically unrealistic bias in favor of flow in coordinate directions. In fact, with such coordinate-direction upstream weighting schemes it is possible for the same numerical method to produce results converging to *different* solutions, depending on the orientation of the finite-difference grid. A considerable amount of current research aims at the construction of upstream weighting methods that exhibit less dependence on the orientation of the coordinate axes. The interested reader may consult Koebbe et al. (1986) for an example of such a method.

To date, finite-element methods for discretizing the oil-reservoir equations have only started to make inroads in industrial simulators. Rather than attempting to survey these wide-ranging results, we refer the reader to a volume in the SIAM series on Frontiers in Applied Mathematics (Ewing, 1983) for a fairly recent overview.

6.6. Problems for Chapter Six.

1. Given the PDE

$$\frac{\partial^2 u}{\partial t^2} + E\frac{\partial^4 u}{\partial x^4} = 0,$$

derive the finite-difference approximation

$$\begin{aligned} &u_i^{n+1} - 2u_i^n + u_i^{n-1} \\ &= -\left(\frac{Ek^2}{h^4}\right)(u_{i+2}^n - 4u_{i+1}^n + 6u_i^n - 4u_{i-1}^n + u_{i-2}^n). \end{aligned}$$

Here, k denotes the time step, and h stands for the spatial grid mesh. Use von Neumann stability analysis to derive the stability criterion $Ek^2/h^4 < \frac{1}{2}$.

2. The aim of this problem is to solve the fourth-order boundary-value problem

$$\begin{aligned} \nabla^4 u + au &= f(x,y) \quad \text{on} \quad \Omega = (0,1)\times(0,1), \\ u(x,y) &= \alpha \quad \text{on} \quad \partial\Omega, \\ \frac{\partial u}{\partial n}(x,y) &= \beta \quad \text{on} \quad \partial\Omega, \end{aligned}$$

where $a > 0$, α, and β are all constants. One finite-difference approximation for the PDE is

$$\begin{aligned} \left(\frac{20}{h^4} + a\right) u_{i,j} &= f(x_i, y_j) \\ &\quad - \frac{1}{h^4}\big[u_{i+2,j} + u_{i-2,j} + u_{i,j+2} + u_{i,j-2} \\ &\quad + 2(u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1}) \\ &\quad - 8(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})\big]. \end{aligned}$$

Here h signifies the grid mesh, assumed uniform. Clearly, whenever (x_i, y_j) is a boundary node, we can fix $u_{i,j} = \alpha$. To handle the normal-derivative condition, we establish a layer of "fictitious" nodes outside the true boundary $\partial\Omega$. Along the left boundary ($x = 0$), for example, Taylor's theorem suggests

$$u_{-1,j} = u_{0,j} - h\frac{\partial u}{\partial n}(x_0, y_j) + \mathcal{O}(h^2),$$

$$u_{1,j} = u_{0,j} + h\frac{\partial u}{\partial n}(x_0, y_j) + \mathcal{O}(h^2),$$

so that, to within $\mathcal{O}(h^2)$,

$$u_{-1,j} = u_{1,j} - 2h\frac{\partial u}{\partial n}(x_0, y_j) = u_{1,j} - 2h\beta.$$

Use this idea to develop a closed system of algebraic equations approximating the original PDE, then write a computer program to solve these equations using the Gauss-Seidel iterative procedure.

3. Consider the ordinary differential equation $du/dt = f(u,t)$, where f is a differentiable function. One implicit finite-difference scheme for this equation is $u_{n+1} - kf(u_{n+1}, t_{n+1}) = u_n$, where subscripts denote time levels and k is the time step. Develop a Newton method for solving this scheme.

4. Give heuristic reasoning showing that, if an iterative scheme for a nonlinear system has asymptotic convergence rate α, then a plot of $\log \|\mathbf{e}^{(m+1)}\|$ versus $\log \|\mathbf{e}^{(m)}\|$, where $\mathbf{e}^{(m)}$ is the error at the m-th iteration, should yield a line with slope α. Use this approach to check the convergence rates of successive substitution and Newton's method for the following equation sets:

(a) $x - \cos x = 0,$

(b) $\dfrac{x^2}{16} - 1 = 0,$

(c) $x^2 + y^2 - x = 0 \quad \text{and} \quad x^2 - y^2 - y = 0.$

5. For the special case of one nonlinear equation in one unknown, say $f(u) = 0$, the method of successive substitution reduces to $u^{(m+1)} = g(u^{(m)})$, where $g(u) = u - f(u)$. Assuming f (and hence g) is differentiable, show that the Lipschitz condition $|g(w) - g(v)| \leq L|w - v|$ holds with $L = \sup_u |g'(u)|$, that is, L can be taken as the least upper bound of $|g'|$. Hint: The mean value theorem states that, if g is continuous on a closed interval $[v, w]$ and differentiable on (v, w), then there is a point $\varsigma \in (v, w)$ such that $g'(\varsigma) = [g(w) - g(v)]/(w - v)$; in other words, $g'(\varsigma)$ equals the average slope of g over the interval.

6. As an alternative to the procedures discussed in Section 6.3, one may consider **predictor-corrector** methods for nonlinear problems. For example, suppose the PDE has the form

$$\frac{\partial^2 u}{\partial x^2} = \mathcal{F}\left(x, t, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial t}\right),$$

where $\mathcal{F}$ is linear in its fifth argument $\partial u/\partial t$. Given a uniform spatial grid of mesh h and a uniform temporal grid of mesh k, a suitable predictor step using finite differences would be

$$\delta_x^2\left(u_i^{n+1/2}\right) = h^2 \mathcal{F}\left(ih, (n+\tfrac{1}{2})k, u_i^n, \frac{u_{i+1}^n - u_{i-1}^n}{2h}, \frac{u_i^{n+1/2} - u_i^n}{k/2}\right).$$

Here, δ_x denotes the central difference operator in the x-direction; see Equation (2.6-12). This step gives a linear algebraic system for the unknown values $u_i^{n+1/2}$ associated with the $n + \frac{1}{2}$ time level. The appropriate corrector step is then

$$\delta_x^2 \left(\frac{u_i^{n+1} - u_i^n}{2} \right)$$
$$= h^2 \mathcal{F} \left(ih, (n + \tfrac{1}{2})k, u_i^{n+1/2}, \frac{u_{i+1}^{n+1/2} - u_{i-1}^{n+1/2}}{2h}, \frac{u_i^{n+1} - u_i^n}{k} \right),$$

which gives a linear problem for the values u_i^{n+1} at the new time level. Construct such a predictor-corrector scheme for the nonlinear heat equation,

$$\frac{\partial}{\partial x} \left[K(u) \frac{\partial u}{\partial x} \right] = \frac{\partial u}{\partial t}.$$

7. Recall that successive overrelaxation (SOR) offers an attractive technique for solving the linear algebraic systems arising from linear elliptic PDEs. When the PDE is nonlinear, its algebraic analog will be, too. **Nonlinear overrelaxation** furnishes an extension of the standard SOR procedure based on Newton's method. Given a system of nonlinear equations $f_i(u_1, \ldots, u_n) = 0, i = 1, \ldots, n$, we perform a set of iterations, each involving a sequential pass through the list $u_1, \ldots, u_n$ of unknowns:

$$u_i^{(m+1)} = u_i^{(m)} - \omega \frac{f_i \left(u_1^{(m+1)}, \ldots, u_{i-1}^{(m+1)}, u_i^{(m)}, \ldots, u_n^{(m)} \right)}{\frac{\partial f_i}{\partial x_i} \left(u_1^{(m+1)}, \ldots, u_{i-1}^{(m+1)}, u_i^{(m)}, \ldots, u_n^{(m)} \right)}.$$

Here, $\omega \in [1, 2]$ is an overrelaxation parameter. Formulate a nonlinear overrelaxation scheme for the nonlinear Poisson problem

$$\nabla^2 u = u^2 \quad \text{on} \quad \Omega = (0, 1) \times (0, 1),$$

$$u(x, 0) = u(x, 1) = 1, \quad u(0, y) = u(1, y) = 0.$$

Write a computer program implementing the scheme.

8. For oil reservoirs in which only oil and water are present, the average pressure $\bar{p} = (p_O + p_W)/2$ and the capillary pressure p_{COW} furnish alternative independent unknowns to the actual fluid pressures p_O and p_W. Reformulate the SS procedure for the black-oil equations, outlined in Section 6.5, in terms of $\bar{p}$ and p_{COW}.

9. The PDE

$$\frac{\partial}{\partial x}\left(K\frac{\partial u}{\partial x}\right) = 0, \quad K > 0,$$

admits a "factored" form,

$$K^{-1}v + \frac{\partial u}{\partial x} = 0,$$
$$\frac{\partial v}{\partial x} = 0,$$

that serves as a starting point for mixed finite-element schemes like those mentioned for fourth-order equations in Section 6.2. Given a uniform grid $x_0 < \ldots < x_N$ of mesh h, suppose we adopt a piecewise constant trial function $\hat{u} = \sum_{i=1}^{N} u_i c_i(x)$ for u. Here,

$$c_i(x) = \begin{cases} 1, & \text{if } x_{i-1} \le x \le x_i, \\ 0, & \text{otherwise.} \end{cases}$$

Let us also adopt a piecewise linear trial function $\hat{v} = \sum_{i=1}^{N} v_i \ell_i(x)$ for v, where $\ell_i(x)$ is the usual Lagrange piecewise linear basis function. The appropriate Galerkin equations have the forms

$$\int \left(K^{-1}\hat{v} + \frac{\partial \hat{u}}{\partial x}\right) \ell_j \, dx = 0,$$
$$\int \frac{\partial \hat{v}}{\partial x} c_j \, dx = 0.$$

Develop the matrix equation for this system with, say, $N = 4$. For simplicity, assume the Neumann boundary conditions $v(x_0) = v(x_N) = 0$, although this leads to a singular equation set with no unique solution. Hint: Use integration by parts where appropriate. For an extension of this approach to two space dimensions, see Allen et al. (1985).

10. Using the factoring described in Problem 9, the PDE

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(K\frac{\partial u}{\partial x}\right), \quad K > 0,$$

can be written as a first-order system,

$$v + K\frac{\partial u}{\partial x} = 0,$$

$$\frac{\partial u}{\partial t} + \frac{\partial v}{\partial x} = 0.$$

assign unknown approximate values of v to the nodes, as in $v_i^n \simeq v(ih, nk)$, and we associate approximate values of u with the element midpoints, as in $u_i^n \simeq u((i+\frac{1}{2})h, nk)$. Thus we can derive the discrete analogs

$$u_i^{n+1} - u_i^n = -\frac{k}{h}(v_i^n - v_{i-1}^n),$$
$$v_i^{n+1} = -\frac{K}{h}(u_{i+1}^{n+1} - u_i^{n+1}).$$

Use von Neumann stability analysis to derive a stability criterion for this scheme. Hint: Let u_i^n, v_i^n have typical Fourier components $\xi^n \exp(\hat{\imath} j h)$, $\eta^n \exp(\hat{\imath} \ell h)$, respectively, and find an expression for η^n in terms of ξ^n.

6.7. References.

Allen, M.B., *Collocation Techniques for Modeling Compositional Flows in Oil Reservoirs*, Berlin: Springer-Verlag, 1984.

Allen, M.B., "Why upwinding is reasonable," in J.P. Laible et al., Eds., *Proceedings of the Fifth International Conference on Finite Elements in Water Resources*, June 18-22, 1984, Burlington, Vermont, Berlin: Springer-Verlag, 1984, pp. 13-23.

Allen, M.B., Ewing, R.E., and Koebbe, J.V., "Mixed finite-element methods for computing groundwater velocities," *Num. Methods Partial Differential Equations 3*(1985), 195-207.

Allen, M.B. and Murphy, C.L., "A finite-element collocation model for variably saturated flows in porous media," *Num. Methods Partial Differential Equations, 3* (1985), 229-239.

Allen, M.B. and Murphy, C.L., "A finite-element collocation method for variably saturated flow in two space dimensions," *Water Resour. Res., 22:11* (1986), 1537-1542.

Aziz, K. and Settari, A., *Petroleum Reservoir Simulation*, London: Applied Science Publishers, 1979.

Birkhoff, G. and Lynch, R.E., *Numerical Solution of Elliptic Problems*, Philadelphia: SIAM, 1984.

Breitenbach, E.A., Thurnau, D.H., and van Poolen, H.K., "Solution of the immiscible fluid flow simulation equation," *Soc. Pet. Eng. J.* (1969), 155-169.

Buckley, S.E. and Leverett, M.C., "Mechanism of fluid displacement in sands," *Trans. AIME., 146* (1942), 107-116.

Carey, G.F. and Oden, J.T., *Finite Elements: A Second Course,* Englewood Cliffs, New Jersey: Prentice-Hall, 1983.

Clough, R.W. and Tocher, J.L., "Finite element stiffness matrices for analysis of plate bending," in *Proceedings, Conference on Matrix Methods in Structural Mechanics*, AFFDL, TR-66-80, Wright-Patterson Air Force Base, Ohio, 1966, 15-26.

Coats, K., "An equation-of-state compositional model," *Soc. Pet. Eng. J.* (1980), 363-376.

Coats, K., Nielsen, R.L., Terhune, M.H., and Weber, A.G., "Simulation of three-dimensional, two-phase flow in oil and gas reservoirs," *Soc. Pet. Eng. J.* (1967), 377-388.

Douglas, J., Peaceman, D.W, and Rachford, H.H., "A method for calculating multidimensional immiscible displacement," *Trans. AIME, 216* (1959), 297-306.

Ewing, R.E., Ed., *The Mathematics of Reservoir Simulation,* Philadelphia: SIAM, 1983.

Griffiths, D.F. and Mitchell, A.R., "Nonconforming elements," in D.F. Griffiths, ed., *The Mathematical Basis of Finite Element Methods*, Oxford: Clarendon Press, 1984, pp. 41-70.

Koebbe, J.V., Ewing, R.E., and Lagnado, R.P., "Accurate velocity weighting techniques," presented at the Second Wyoming Symposium on Enhanced Oil Recovery, Casper, Wyoming, May 15-16, 1986.

Lapidus, L. and Pinder, G.F., *Numerical Methods for Partial Differential Equations in Science and Engineering*, New York: John Wiley and Sons, 1982.

Ortega, J. and Rheinboldt, W.C., *Iterative Solution of Nonlinear Equations in Several Variables*, New York: Academic Press, 1970.

Peaceman, D.W., *Fundamentals of Numerical Reservoir Simulation*, Amsterdam: Elsevier, 1977.

Safai, N.M., "Simulation of Saturated and Unsaturated Deformable Porous Media," Ph.D. Thesis, Department of Civil Engineering, Princeton University, Princeton, New Jersey, 1977.

Sheldon, J.W., Zondek, B., and Cardwell, W.T., "One-dimensional, incompressible, non-capillary two-phase flow in a porous medium," *Trans. AIME, 216* (1959), 290-296.

Stone, H.L. and Garder, A.O., "Analysis of gas-cap or dissolved-gas reservoirs," *Trans. AIME, 222* (1961), 92-104.

Strang, G. and Fix, G.F., *An Analysis of the Finite Element Method*, Englewood Cliffs, New Jersey: Prentice-Hall, 1973.

This Appendix reviews some of the basic facts about vectors and tensors used in our discussions of mechanics. We begin by defining notation and reviewing the algebraic operations frequently encountered in the text. Next, we briefly treat the differentiation of vectors and tensors. Finally, we review some integral theorems used throughout the text.

Notation.

We shall restrict our attention to three-dimensional Euclidean space, and in what follows we assume a Cartesian coordinate system with axes labeled $1, 2, 3$. In this framework, a **quantity of rank** n is a collection of 3^n real numbers or variables, called **entries**, each of which is labeled by n indices. The cases encountered in most treatments of mechanics are $n = 0, 1, 2$. When $n = 0$, the quantity in question is a single real number or variable s, that is, a **scalar**. Pressure, for example, is a scalar. When $n = 1$, the quantity in question is a **vector**, $\mathbf{V}$. We may write a vector explicitly as a column array,

$$\mathbf{V} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}.$$

Velocity, for example, is an ordered triple of components along the three coordinate axes and hence is a vector. Informally, a vector is a quantity having both direction and magnitude. When $n = 2$, the quantity in question is a **tensor**, $\mathbf{T}$. We can list the entries of a tensor in array form as follows:

$$\mathbf{T} = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix}.$$

Stress is perhaps the prototypical example of a tensor. Its entries represent the three components of momentum flux across planes perpendicular to each of the three coordinate axes, as explained in Chapter One.

We use two notations to signify these different types of quantities. One is **direct** notation, in which we write scalars in ordinary type, s; vectors in boldface, $\mathbf{V}$, and tensors in boldface sans-serif type, T. This notation is the one we emphasize in the main text of the book. The other notation is **index** notation, in which we write quantities of all ranks in ordinary type, specifying the rank of a quantity through the number of free (unrepeated, nonnumerical) indices. Thus a scalar appears as s, a vector as V_i, and a tensor as T_{ij}. Observe that the variable indices i, j, and so forth, appearing in the index notation stand, not for particular numerical values, but for generic values. Thus in V_i the index i represents at once all of its possible values 1, 2, and 3, so that V_i is actually shorthand for the ordered triple whose entries are V_1, V_2, and V_3. Compared with index notation, direct notation seems better to facilitate physical intuition, and therefore the text exhibits a bias in its favor. However, index notation has greater flexibility and is less vulnerable to ambiguity. In what follows we shall outline the basic algebraic and differential operations using both notations.

Basic algebra.

Addition of two scalars r and s is the usual operation, $r + s$. To compute the sum of two vectors $\mathbf{V}$ and $\mathbf{W}$, we add entries having the same index:

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} + \begin{bmatrix} W_1 \\ W_2 \\ W_3 \end{bmatrix} = \begin{bmatrix} V_1 + W_1 \\ V_2 + W_2 \\ V_3 + W_3 \end{bmatrix} = \begin{cases} \mathbf{V} + \mathbf{W} & \text{(direct notation)} \\ V_i + W_i & \text{(index notation).} \end{cases}$$

Addition of two tensors T and U is similar:

$$\begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} + \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ U_{21} & U_{22} & U_{23} \\ U_{31} & U_{32} & U_{33} \end{bmatrix}$$

$$= \begin{bmatrix} T_{11} + U_{11} & T_{12} + U_{12} & T_{13} + U_{13} \\ T_{21} + U_{21} & T_{22} + U_{22} & T_{23} + U_{23} \\ T_{31} + U_{31} & T_{32} + U_{32} & T_{33} + U_{33} \end{bmatrix}$$

$$= \begin{cases} \mathsf{T} + \mathsf{U} & \text{(direct notation)} \\ T_{ij} + U_{ij} & \text{(index notation).} \end{cases}$$

One can easily see that $\mathbf{V} + \mathbf{W} = \mathbf{W} + \mathbf{V}$, so that vector addition is commutative, and that $(\mathbf{V} + \mathbf{W}) + \mathbf{Y} = \mathbf{V} + (\mathbf{W} + \mathbf{Y})$, so that it is associative as well. Corresponding properties hold for tensor addition.

To **multiply** a vector $\mathbf{V}$ by a scalar s, simply multiply the entries of the vector by s:

$$s\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} sV_1 \\ sV_2 \\ sV_3 \end{bmatrix} = \begin{cases} s\mathbf{V} & \text{(direct notation)} \\ sV_i & \text{(index notation).} \end{cases}$$

The rule for tensors is similar:

$$s = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} = \begin{bmatrix} sT_{11} & sT_{12} & sT_{13} \\ sT_{21} & sT_{22} & sT_{23} \\ sT_{31} & sT_{32} & sT_{33} \end{bmatrix}$$

$$= \begin{cases} s\mathbf{T} & \text{(direct notation)} \\ sT_{ij} & \text{(index notation).} \end{cases}$$

Multiplication by scalars is **commutative** in the sense that $s\mathbf{V} = \mathbf{V}s$, and it is **distributive** in the following two senses:

$$(r+s)\mathbf{V} = r\mathbf{V} + s\mathbf{V},$$

$$r(\mathbf{V}+\mathbf{W}) = r\mathbf{V} + r\mathbf{W}.$$

Similar properties hold for tensors. We can use addition and multiplication by scalars to decompose vectors into their components along the three Cartesian axes. Let us denote the **usual basis** for three-dimensional space by

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Then a vector $\mathbf{V}$ decomposes as follows:

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \sum_{i=1}^{3} V_i \mathbf{e}_i.$$

Transposition.

If we consider vectors to be column arrays, then the **transpose** of a vector $\mathbf{V}$ is, formally, a row array:

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}^{\top} = [V_1, V_2, V_3].$$

For tensors, transposition amounts to a flip across the diagonal:

$$\begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix}^{\top} = \begin{bmatrix} T_{11} & T_{21} & T_{31} \\ T_{12} & T_{22} & T_{32} \\ T_{13} & T_{23} & T_{33} \end{bmatrix}$$

$$= \begin{cases} \mathbf{T}^{\top} & \text{(direct notation)} \\ T_{ij}^{\top} = T_{ji} & \text{(index notation).} \end{cases}$$

A tensor $\mathbf{T}$ is **symmetric** if $\mathbf{T} = \mathbf{T}^{\top}$; it is **antisymmetric** if $\mathbf{T} = -\mathbf{T}^{\top}$. Notice that the diagonal entries of an antisymmetric tensor must all be zero.

Dot products.

The **dot product** of two vectors $\mathbf{V}$ and $\mathbf{W}$ is the following:

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}^{\top} \cdot \begin{bmatrix} W_1 \\ W_2 \\ W_3 \end{bmatrix} = [V_1, V_2, V_3] \begin{bmatrix} W_1 \\ W_2 \\ W_3 \end{bmatrix} = \sum_{i=1}^{3} V_i W_i.$$

By extension of the Pythagorean theorem, we can use the dot product to define the **Euclidean length** of any vector $\mathbf{V}$:

$$\|\mathbf{V}\|_2 = (\mathbf{V} \cdot \mathbf{V})^{1/2} = \sqrt{\sum_{i=1}^{3} V_i V_i}.$$

While it may not be immediately apparent, in the last two sums the notation "$\sum_{i=1}^{3}$" is actually superfluous. Since sums of this sort appear quite frequently in vector and tensor analysis, it is common to adopt a convention that the repetition of a variable index such as i or j in a single term implies summation over the values $1, 2, 3$ for that index. This rule is called the **Einstein summation convention**, and it affords great notational convenience with very little risk of ambiguity. Using this convention, we may write the dot product of two vectors as follows: $\mathbf{V}$ and $\mathbf{W}$ in index notation as $V_i W_i$, the symbol $\sum_{i=1}^{3}$ being understood since the index i appears repeated. Notice, however, that while $V_i W_i$ is a quantity of rank 0, $V_i W_j$ has no repeated indices and is therefore a quantity of rank 2, sometimes called the **dyadic** (or **tensor**) **product** of $\mathbf{V}$ and $\mathbf{W}$:

$$\begin{bmatrix} V_1W_1 & V_1W_2 & V_1W_3 \\ V_2W_1 & V_2W_2 & V_2W_3 \\ V_3W_1 & V_3W_2 & V_3W_3 \end{bmatrix} = \begin{cases} \mathbf{VW} & \text{(direct notation)} \\ V_i W_j & \text{(index notation).} \end{cases}$$

Using the Einstein summation convention, the dot product of vectors $\mathbf{V}$ and $\mathbf{W}$ is as follows:

$$\sum_{i=1}^{3} V_i W_i = \begin{cases} \mathbf{V} \cdot \mathbf{W} & \text{(index notation)} \\ V_i W_i & \text{(direct notation).} \end{cases}$$

This operation is commutative, and it is distributive:

$$\mathbf{V} \cdot (\mathbf{W} + \mathbf{Y}) = \mathbf{V} \cdot \mathbf{W} + \mathbf{V} \cdot \mathbf{Y}.$$

If $\mathbf{V}$ and $\mathbf{W}$ are vectors whose magnitudes are $\|\mathbf{V}\|_2$ and $\|\mathbf{W}\|_2$, respectively, and whose directions in three-space differ by an angle θ, then $\mathbf{V} \cdot \mathbf{W}$ is a scalar whose magnitude is $\|\mathbf{V}\|_2 \|\mathbf{W}\|_2 \cos\theta$. This formula implies that two vectors $\mathbf{V}$ and $\mathbf{W}$ are orthogonal precisely when $\mathbf{V} \cdot \mathbf{W} = 0$.

The dot product of a vector and a tensor is analogous to matrix multiplication:

$$\begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \sum_{j=1}^{3} \begin{bmatrix} T_{1j} V_j \\ T_{2j} V_j \\ T_{3j} V_j \end{bmatrix}$$

$$= \begin{cases} \mathbf{T} \cdot \mathbf{V} & \text{(direct notation)} \\ T_{ij} V_j & \text{(index notation).} \end{cases}$$

Alternatively, we can compute

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}^{\top} \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} = \sum_{i=1}^{3} \begin{bmatrix} V_i T_{i1} \\ V_i T_{i2} \\ V_i T_{i3} \end{bmatrix}$$

$$= \begin{cases} \mathbf{V}^{\top} \cdot \mathbf{T} & \text{(direct notation)} \\ V_i T_{ij} & \text{(index notation).} \end{cases}$$

Notice that $\mathbf{T} \cdot \mathbf{V} = \mathbf{V}^{\top} \cdot \mathbf{T}$ for general $\mathbf{V}$ only if $\mathbf{T}$ is symmetric.

Finally, given two tensors $\mathbf{T}$ and $\mathbf{U}$, we can from the **double-dot product**, or **contraction**, as follows:

$$\begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} : \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ U_{21} & U_{22} & U_{23} \\ U_{31} & U_{32} & U_{33} \end{bmatrix}$$

$$= \sum_{i=1}^{3} \sum_{j=1}^{3} T_{ij} U_{ij} = \begin{cases} \mathbf{T} : \mathbf{U} & \text{(direct notation)} \\ T_{ij} U_{ij} & \text{(index notation).} \end{cases}$$

It is easy to show that the double-dot product is commutative: $\mathbf{T} : \mathbf{U} = \mathbf{U} : \mathbf{T}$.

The Kronecker and Levi-Civita symbols.

Two notational tools are quite useful in defining higher algebraic operations among vectors and tensors. The first is the **Kronecker symbol**, defined as follows:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j. \end{cases}$$

This symbol, especially useful in index notation, corresponds in direct notation to the **identity tensor**

$$\mathbf{1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The second notational tool is the **Levi-Civita symbol**. This symbol has rank 3, and its value depends on whether the indices (i, j, k) form an even permutation of $(1, 2, 3)$, an odd permutation of $(1, 2, 3)$, or a combination with one index repeated:

$$\epsilon_{ijk} = \begin{cases} 1 & \text{if } (i,j,k) = (1,2,3), (2,3,1), \text{ or } (3,1,2), \\ -1 & \text{if } (i,j,k) = (1,3,2), (3,2,1), \text{ or } (2,1,3), \\ 0 & \text{if any index is repeated.} \end{cases}$$

Trace and determinant.

The **trace** of a tensor is the sum of its diagonal entries:

$$\text{tr} \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} = T_{11} + T_{22} + T_{33}$$

$$= \begin{cases} \text{tr}\mathbf{T} & \text{(direct notation)} \\ \delta_{ij} T_{ij} = T_{ii} & \text{(index notation).} \end{cases}$$

The **determinant** of a tensor is defined as the determinant of the matrix having the same entries:

$$\det \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} = T_{11}(T_{22}T_{33} - T_{23}T_{32}) - T_{12}(T_{21}T_{33} - T_{23}T_{31})$$

$$+T_{13}(T_{21}T_{32} - T_{22}T_{31}) = \begin{cases} \det \mathbf{T} & \text{(direct notation)} \\ \epsilon_{ijk} T_{1i} T_{2j} T_{3k} & \text{(index notation).} \end{cases}$$

One can show that both the trace and the determinant of a tensor remain invariant under transformations to new orthogonal coordinate systems.

Cross products.

The most familiar cross product is that defined for two vectors:

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \times \begin{bmatrix} W_1 \\ W_2 \\ W_3 \end{bmatrix} = \begin{bmatrix} V_2W_3 - V_3W_2 \\ V_3W_1 - V_1W_3 \\ V_1W_2 - V_2W_1 \end{bmatrix}$$

$$= \begin{cases} \mathbf{V} \times \mathbf{W} & \text{(direct notation)} \\ \epsilon_{ijk} V_j W_k & \text{(index notation).} \end{cases}$$

It is easy to prove, using the Levi-Civita symbol, that the cross product is **anticommutative:** $\mathbf{V} \times \mathbf{W} = -\mathbf{W} \times \mathbf{V}$. Geometrically, $\mathbf{V} \times \mathbf{W}$ is a vector perpendicular to both $\mathbf{V}$ and $\mathbf{W}$. If $\mathbf{V}$ and $\mathbf{W}$ stand at an angle θ to each other, the magnitude of $\mathbf{V} \times \mathbf{W}$ is $\|\mathbf{V}\|_2 \|\mathbf{W}\|_2 \sin\theta$, which is twice the area of the parallelogram formed by $\mathbf{V}$, $\mathbf{W}$, and $\mathbf{V} - \mathbf{W}$.

It is also possible, though somewhat uncommon, to define cross products using tensors. For example, we can form the cross product of a vector and a tensor, yielding another tensor:

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \times \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix}$$

$$= \begin{bmatrix} V_2T_{13} - V_3T_{12} & V_2T_{23} - V_3T_{22} & V_2T_{33} - V_3T_{32} \\ V_3T_{11} - V_1T_{13} & V_3T_{21} - V_1T_{23} & V_3T_{31} - V_1T_{33} \\ V_1T_{12} - V_2T_{11} & V_1T_{22} - V_2T_{21} & V_1T_{32} - V_2T_{31} \end{bmatrix}$$

$$= \begin{cases} \mathbf{V} \times \mathbf{T} & \text{(direct notation)} \\ \epsilon_{ijk} V_j T_{mk} & \text{(index notation).} \end{cases}$$

Similarly, we can define the cross product of two tensors to be the following scalar:

$$\begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} \times \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ U_{21} & U_{22} & U_{23} \\ U_{31} & U_{32} & U_{33} \end{bmatrix}$$

$$= \sum_{m=1}^{3} \begin{bmatrix} T_{2m}U_{m3} - T_{3m}U_{m2} \\ T_{3m}U_{m1} - T_{1m}U_{m3} \\ T_{1m}U_{m2} - T_{2m}U_{m1} \end{bmatrix} = \begin{cases} \mathbf{T} \times \mathbf{U} & \text{(direct notation)} \\ \epsilon_{ijk} T_{jm} U_{mk} & \text{(index notation).} \end{cases}$$

Table A-1 lists some useful algebraic identities for scalars, vectors, and tensors.

Partial differentiation.

The fundamental differential operator in vector and tensor analysis is the **del** or **nabla** operator, denoted in Cartesian coordinates by

$$\nabla = \begin{bmatrix} \partial/\partial x_1 \\ \partial/\partial x_2 \\ \partial/\partial x_3 \end{bmatrix}.$$

The vector notation used here, known as Gibbs' notation, is less appropriate for use with non-Cartesian coordinates; however, it is useful mnemonically.

The operator ∇ can act on scalar functions, vector functions, and tensor functions, provided their component functions are differentiable. In the case of a scalar function s, we can form the **gradient** ∇s, which is a function of rank 1:

$$\begin{bmatrix} \partial s/\partial x_1 \\ \partial s/\partial x_2 \\ \partial s/\partial x_3 \end{bmatrix} = \begin{cases} \nabla s & \text{(direct notation)} \\ \partial s/\partial x_i & \text{(index notation).} \end{cases}$$

For a vector function $\mathbf{V}$ there are more possibilities. Formally taking the dot product of ∇ with $\mathbf{V}$ gives the **divergence** of the vector function,

$$\frac{\partial V_1}{\partial x_1} + \frac{\partial V_2}{\partial x_2} + \frac{\partial V_3}{\partial x_3} = \begin{cases} \nabla \cdot \mathbf{V} & \text{(direct notation)} \\ \partial V_i/\partial x_i & \text{(index notation).} \end{cases}$$

This function is a quantity of rank 0. Similarly, it is possible to take the formal cross product of ∇ and $\mathbf{V}$, producing the **curl** of $\mathbf{V}$:

$$\begin{bmatrix} \partial V_3/\partial x_2 - \partial V_2/\partial x_3 \\ \partial V_1/\partial x_3 - \partial V_3/\partial x_1 \\ \partial V_2/\partial x_1 - \partial V_1/\partial x_2 \end{bmatrix} = \begin{cases} \nabla \times \mathbf{V} & \text{(direct notation)} \\ \epsilon_{ijk}\partial V_k/\partial x_j & \text{(index notation).} \end{cases}$$

The curl is clearly a function of rank 1. To get a function of rank 2 we can form the **vector gradient** of $\mathbf{V}$:

$$\begin{bmatrix} \partial V_1/\partial x_1 & \partial V_1/\partial x_2 & \partial V_1/\partial x_3 \\ \partial V_2/\partial x_1 & \partial V_2/\partial x_2 & \partial V_2/\partial x_3 \\ \partial V_3/\partial x_1 & \partial V_3/\partial x_2 & \partial V_3/\partial x_3 \end{bmatrix} = \begin{cases} \nabla \mathbf{V} & \text{(direct notation)} \\ \partial V_j/\partial x_i & \text{(index notation).} \end{cases}$$

In the special case when the vector function $\mathbf{V}$ is the gradient of a scalar, say ∇s, taking the divergence of $\mathbf{V}$ yields

$$\frac{\partial^2 s}{\partial x_1^2} + \frac{\partial^2 s}{\partial x_2^2} + \frac{\partial^2 s}{\partial x_3^2} = \begin{cases} \nabla \cdot (\nabla s) = \nabla^2 s & \text{(direct notation)} \\ \partial^2 s/\partial x_i \partial x_i & \text{(index notation).} \end{cases}$$

TABLE A-1. Algebraic identities for vectors and tensors.

Dot Products

$\mathbf{V}_1 \cdot \mathbf{V}_2 = \mathbf{V}_2 \cdot \mathbf{V}_1$

$\mathbf{V}_1 \cdot (\mathbf{V}_2 + \mathbf{V}_3) = \mathbf{V}_1 \cdot \mathbf{V}_2 + \mathbf{V}_1 \cdot \mathbf{V}_3$

$\mathbf{T} : \mathbf{U} = \mathbf{U} : \mathbf{T}$

Cross products

$\mathbf{V}_1 \times \mathbf{V}_2 = -\mathbf{V}_2 \times \mathbf{V}_1$

$\mathbf{V}_1 \times (\mathbf{V}_2 + \mathbf{V}_3) = \mathbf{V}_1 \times \mathbf{V}_2 + \mathbf{V}_1 \times \mathbf{V}_3$

$\mathbf{V}_1 \times (\mathbf{V}_2 \times \mathbf{V}_3) = (\mathbf{V}_3 \cdot \mathbf{V}_1)\mathbf{V}_2 - (\mathbf{V}_1 \cdot \mathbf{V}_2)\mathbf{V}_3$ (vector triple product)

$\mathbf{V} \times \mathbf{V} = 0$

$\mathbf{V}_1 \cdot (\mathbf{V}_2 \times \mathbf{V}_3) = \mathbf{V}_2 \cdot (\mathbf{V}_3 \times \mathbf{V}_1) = \mathbf{V}_3 \cdot (\mathbf{V}_1 \times \mathbf{V}_2)$

$(\mathbf{V}_1 \times \mathbf{V}_2) \cdot (\mathbf{V}_3 \times \mathbf{V}_4) = (\mathbf{V}_1 \cdot \mathbf{V}_3)(\mathbf{V}_2 \cdot \mathbf{V}_4) - (\mathbf{V}_1 \cdot \mathbf{V}_4)(\mathbf{V}_2 \cdot \mathbf{V}_3)$

$(\mathbf{V}_1 \times \mathbf{V}_2) \times (\mathbf{V}_3 \times \mathbf{V}_4) = [(\mathbf{V}_1 \times \mathbf{V}_2) \cdot \mathbf{V}_4]\mathbf{V}_3 - [(\mathbf{V}_1 \times \mathbf{V}_2) \cdot \mathbf{V}_3]\mathbf{V}_4$

The operator ∇^2 appears frequently in mechanics; it is called the **Laplace operator.**

Finally, given a tensor function $\mathbf{T}$ we can form its divergence:

$$\sum_{i=1}^{3} \begin{bmatrix} \partial T_{i1}/\partial x_i \\ \partial T_{i2}/\partial x_i \\ \partial T_{i3}/\partial x_i \end{bmatrix} = \begin{cases} \nabla \cdot \mathbf{T} & \text{(direct notation)} \\ \partial T_{ij}/\partial x_i & \text{(index notation).} \end{cases}$$

This derivative is a function of rank 1.

Table A-2 contains some useful differential identities using the operator ∇.

Integral theorems.

There are three important integral theorems that find application in various parts of the book. While we review the basic statements of these theorems here, a formal treatment of them would require attention to quite a few technicalities that lie beyond the scope of this text. We refer the interested reader to Crowell, Williamson, and Trotter, *Calculus of Vector Functions* (1972), cited in the references to Chapter One.

The first of the theorems, the **Gauss** or **divergence theorem**, is essentially a generalization of the fundamental theorem of calculus to volume integrals. Suppose $\mathcal{V}$ is a connected region in three-dimensional Euclidean space having a smooth, closed boundary $\partial\mathcal{V}$. Suppose further that $\partial\mathcal{V}$ is orientable, so that it is possible to choose an unambiguous outward direction on $\partial\mathcal{V}$. Let $\mathbf{n}$ stand for a vector function defined on the points $\mathbf{x}$ of $\partial\mathcal{V}$ such that $\mathbf{n}(\mathbf{x})$ has unit length, is normal to the plane tangent to $\partial\mathcal{V}$ at $\mathbf{x}$, and points in the outward direction as drawn in Figure A-1. Then if $\mathbf{V}$ is a continuously differentiable vector-valued function defined on $\mathcal{V}$,

$$\int_{\mathcal{V}} \nabla \cdot \mathbf{V} \, d\mathbf{x} = \oint_{\partial\mathcal{V}} \mathbf{V} \cdot \mathbf{n} \, d\mathbf{x}.$$

The integral on the left in this equation is a volume integral over $\mathcal{V}$; the integral on the right is a surface integral over the closed surface $\partial\mathcal{V}$.

The second integral theorem is sometimes called **Green's theorem**. This theorem is a three-dimensional analog of integration by parts in that it allows one to shift differential operations in a volume integral while incurring a contribution from the boundary of the volume. Let $\mathcal{V}$, $\partial\mathcal{V}$, and $\mathbf{n}$ be as above, and suppose f and g are two scalar functions that are sufficiently smooth to permit all of the differential operations called for below. Then

$$\int_{\mathcal{V}} f \nabla^2 g \, d\mathbf{x} = -\int_{\mathcal{V}} (\nabla f) \cdot (\nabla g) d\mathbf{x} + \oint_{\partial\mathcal{V}} f \, \nabla g \cdot \mathbf{n} \, d\mathbf{x}.$$

TABLE A-2. Differential identities for vectors and tensors.

Gradients

$$\nabla(r+s) = \nabla r + \nabla s$$

$$\nabla(rs) = r\nabla s + s\nabla r$$

$$\nabla(\mathbf{V}\cdot\mathbf{W}) = \mathbf{V}\cdot\nabla\mathbf{W} + \mathbf{W}\cdot\nabla\mathbf{V} + \mathbf{V}\times\nabla\times\mathbf{W} + \mathbf{W}\times\nabla\times\mathbf{V}$$

Divergences

$$\nabla\cdot(\mathbf{V}+\mathbf{W}) = \nabla\cdot\mathbf{V} + \nabla\cdot\mathbf{W}$$

$$\nabla\cdot(s\mathbf{V}) = \mathbf{V}\cdot\nabla s + s\nabla\cdot\mathbf{V}$$

$$\nabla\cdot(\mathbf{V}\times\mathbf{W}) = \mathbf{W}\cdot\nabla\times\mathbf{V} - \mathbf{V}\cdot\nabla\times\mathbf{W}$$

$$\nabla\cdot(\nabla\times\mathbf{V}) = 0$$

$$\nabla\cdot(s\mathbf{1}) = \nabla s \quad (\mathbf{1} = \text{identity tensor})$$

$$\nabla\cdot(\mathbf{T}\cdot\mathbf{V}) = \mathbf{T} : \nabla\mathbf{V} + \mathbf{V}\cdot(\nabla\cdot\mathbf{T})$$

$$\nabla\cdot(\mathbf{V}\times\mathbf{T}) = \mathbf{V}\times(\nabla\cdot\mathbf{T}) + (\nabla\mathbf{V})\times\mathbf{T}$$

Curls

$$\nabla\times(\mathbf{V}+\mathbf{W}) = \nabla\times\mathbf{V} + \nabla\times\mathbf{W}$$

$$\nabla\times(s\mathbf{V}) = s\nabla\times\mathbf{V} - \mathbf{V}\times\nabla s$$

$$\nabla\times(\mathbf{V}\times\mathbf{W}) = (\nabla\cdot\mathbf{W})\mathbf{V} - (\nabla\cdot\mathbf{V})\mathbf{W} + \mathbf{W}\cdot\nabla\mathbf{V} - \mathbf{V}\cdot\nabla\mathbf{W}$$

$$\nabla\times\nabla\times\mathbf{V} = \nabla(\nabla\cdot\mathbf{V}) - \nabla^2 V$$

$$\nabla\times\nabla s = 0$$

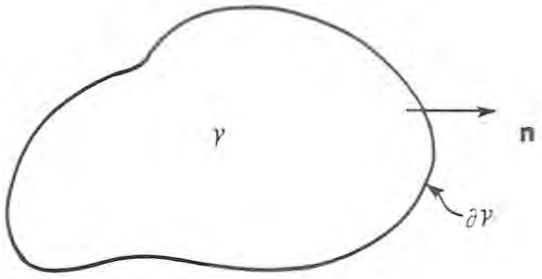

Figure A-1. A connected region $\mathcal{V}$ with smooth, orientable boundary $\partial\mathcal{V}$ and unit outward normal vector $\mathbf{n}$.

The last integral theorem we shall review tells how to change variables in a volume integral. Suppose ϕ is an integrable scalar function defined on three-dimensional Euclidean space, and consider a continuously differentiable change of variables $\mathbf{x} = \mathbf{f}(\mathbf{u})$ mapping points in a region $\mathcal{V}$ in $\mathbf{u}$-coordinates to points in a region $f(\mathcal{V})$ in $\mathbf{x}$-coordinates, as drawn in Figure A-2. Denote by $\mathbf{J}$ the Jacobian matrix of the mapping $\mathbf{f}$, namely,

$$\mathbf{J} = \begin{bmatrix} \partial x_1/\partial u_1 & \partial x_1/\partial u_2 & \partial x_1/\partial u_3 \\ \partial x_2/\partial u_1 & \partial x_2/\partial u_2 & \partial x_2/\partial u_3 \\ \partial x_3/\partial u_1 & \partial x_3/\partial u_2 & \partial x_3/\partial u_3 \end{bmatrix}.$$

Then the change of variables theorem relates the integral of ϕ over volumes in $\mathbf{x}$-coordinates to integrals over volumes in $\mathbf{u}$-coordinates as follows:

$$\int_{f(\mathcal{V})} \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathcal{V}} \phi(\mathbf{f}(\mathbf{u}))\, |\det(\mathbf{J})| d\mathbf{x}.$$

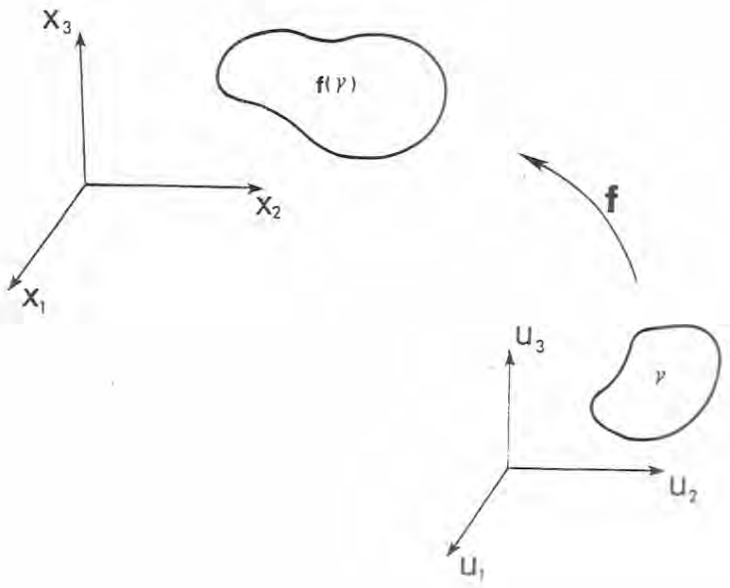

Figure A-2. Schematic of the relationship between a volume $\mathcal{V}$ in **u**-coordinates and the same volume as it appears under the change of variables $\mathbf{x} = \mathbf{f}(\mathbf{u})$.