Chapter 6

Nonlinear Problems

The preceding chapters presented multigrid methods applied to linear problems. Because of their iterative nature, multigrid ideas should be effective on nonlinear problems, where some form of iteration is usually imperative. Just as multigrid methods were developed to improve classical linear relaxation methods, we will see that they can also guide us in the use of nonlinear methods.

We begin by clarifying the most significant formal difference between linear and nonlinear systems of equations. Consider a system of nonlinear algebraic equations, $A(\mathbf{u}) = \mathbf{f}$, where $\mathbf{u}, \mathbf{f} \in \mathbf{R}^n$. (The notation $A(\mathbf{u})$, rather than $A\mathbf{u}$, signifies that the operator is nonlinear.) Suppose that $\mathbf{v}$ is an approximation to the exact solution $\mathbf{u}$. It is possible to define the error and the residual for this problem, much as we did earlier: the error is simply $\mathbf{e} = \mathbf{u} - \mathbf{v}$ and the residual is $\mathbf{r} = \mathbf{f} - A(\mathbf{v})$. If we now subtract the original equation, $A(\mathbf{u}) = \mathbf{f}$, from the definition of the residual, we find that

$$A(\mathbf{u}) - A(\mathbf{v}) = \mathbf{r}. \tag{6.1}$$

We are one step from the residual equation that we derived for linear systems in Chapter 2. However, because A is nonlinear, even though $\mathbf{u} - \mathbf{v} = \mathbf{e}$, we *cannot* conclude that $A(\mathbf{u}) - A(\mathbf{v}) = A(\mathbf{e})$. This means that we no longer have a simple linear residual equation, which necessitates changes in the methods that we devise for nonlinear problems. We must now use (6.1) as the residual equation.

It makes sense to review quickly a classical relaxation method for nonlinear systems; it will have a part to play in multigrid methods. Working with the system of nonlinear algebraic equations $A(\mathbf{u}) = \mathbf{f}$, one of the most frequently used methods is *nonlinear Gauss–Seidel* relaxation [16]. Suppose the jth equation of the system can be solved for the jth variable in terms of the other variables $(u_1, u_2, \ldots, u_{j-1}, u_{j+1}, \ldots, u_n)$. We write the result in the abstract form

$$u_j = M_j(u_1, u_2, \ldots, u_{j-1}, u_{j+1}, \ldots, u_n), \quad 1 \le j \le n.$$

Just as with linear systems, Gauss–Seidel relaxation takes the form

$$v_j \leftarrow M_j(v_1, v_2, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n), \quad 1 \le j \le n,$$

where the current value of each component is used on the right side. For cases where this iteration cannot be formed explicitly, we use the same characterization that was used for linear Gauss–Seidel: for each $j = 1, 2, \ldots, n$, set the jth component

of the residual to zero and solve for v_j. Because the residual is $\mathbf{r} = \mathbf{f} - A(\mathbf{v})$, this characterization amounts to solving $(A(\mathbf{v}))_j = \mathbf{f}_j$ for v_j. Equivalently, as in the linear case, we can let $\boldsymbol{\epsilon}_j$ be the jth unit vector, so that the jth step of nonlinear Gauss–Seidel amounts to finding an $s \in \mathbf{R}$ such that

$$(A(\mathbf{v} + s\boldsymbol{\epsilon}_j))_j = f_j, \quad 1 \le j \le n.$$

This is generally a nonlinear scalar equation in the scalar s, which can be solved efficiently with one or two steps of (scalar) Newton's method [16]. When an approximate solution s is found, the jth component is updated by $\mathbf{v} \leftarrow \mathbf{v} + s\boldsymbol{\epsilon}_j$. Updating all n components sequentially in this way constitutes one iteration sweep of nonlinear Gauss–Seidel.

Now, to see how residual equation (6.1) can be used as a basis for multigrid methods, let $\mathbf{v}$ be the current approximation and replace the exact solution $\mathbf{u}$ by $\mathbf{v} + \mathbf{e}$. Then residual equation (6.1) becomes

$$A(\mathbf{v} + \mathbf{e}) - A(\mathbf{v}) = \mathbf{r}. \tag{6.2}$$

Expanding $A(\mathbf{v} + \mathbf{e})$ in a Taylor series (in n variables) about $\mathbf{v}$ and truncating the series after two terms, we have the linear system of equations

$$J(\mathbf{v})\mathbf{e} \;\; = \;\; \mathbf{r}, \tag{6.3}$$

where $J = (\partial A_i/\partial u_j)$ is the $n \times n$ Jacobian matrix. Linear system (6.3) represents an approximation to nonlinear system (6.1). It can be solved for $\mathbf{e}$ and the current approximation $\mathbf{v}$ can be updated by $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{e}$. Iteration of this step is *Newton's method*. But how should linear system (6.3) be solved at each step? One highly recommended option is to use multigrid. Such a combination of Newton's method for the outer iteration and multigrid for the (linear) inner iteration is called *Newton-multigrid*.

Newton-multigrid can be effective, but it does not use multigrid ideas to treat the nonlinearity directly. To do this, we return to the residual equation,

$$A(\mathbf{v} + \mathbf{e}) - A(\mathbf{v}) = \mathbf{r},$$

and use it in the familiar two-grid setting. Suppose we have found an approximation, $\mathbf{v}^h$, to the original fine-grid problem

$$A^h(\mathbf{u}^h) = \mathbf{f}^h\,. \tag{6.4}$$

Proceeding as we did for the linear problem, we now want to use the residual equation on the coarse grid Ω^{2h} to approximate $\mathbf{e}^h$, the error in $\mathbf{v}^h$. Using the above argument, the residual equation on the coarse grid appears as

$$A^{2h}(\mathbf{v}^{2h} + \mathbf{e}^{2h}) - A^{2h}(\mathbf{v}^{2h}) = \mathbf{r}^{2h}, \tag{6.5}$$

where A^{2h} denotes the coarse-grid operator, $\mathbf{r}^{2h}$ is the coarse-grid residual, $\mathbf{v}^{2h}$ is a coarse-grid approximation to $\mathbf{v}^h$, and $\mathbf{e}^{2h}$ is a coarse-grid approximation to $\mathbf{e}^h$. Once $\mathbf{e}^{2h}$ is computed, the fine-grid approximation can be updated by $v^h \leftarrow v^h + I_{2h}^h e^{2h}$.

We have already encountered the coarse-grid residual $\mathbf{r}^{2h}$. Nothing changes here; we simply choose it to be the restriction of the fine-grid residual to the coarse grid:

$$\mathbf{r}^{2h} = I_h^{2h}\mathbf{r}^h = I_h^{2h}(\mathbf{f}^h - A^h(\mathbf{v}^h)).$$

Newton's Method. Perhaps the best known and most important method for solving nonlinear equations is *Newton's method*, which can be derived as follows. Suppose we wish to solve the scalar equation $F(x) = 0$. We expand F in a Taylor series about an initial guess x:

$$F(x+s) = F(x) + sF'(x) + \frac{s^2}{2}F''(\xi),$$

where ξ is between x and $x+s$. If $x+s$ is the solution, then (neglecting the higher-order terms) the series becomes $0 = F(x) + sF'(x)$, from which $s = -F(x)/F'(x)$. Thus, we can update the initial guess using $x \leftarrow x - F(x)/F'(x)$. Newton's method results by iterating this process:

$$x \leftarrow x - \frac{F(x)}{F'(x)}.$$

Newton's method for a system of n nonlinear equations is a straightforward extension of the scalar Newton's method. We write the system in vector form as

$$\mathbf{F}(\mathbf{x}) \equiv \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Expanding in a Taylor series yields

$$\mathbf{F}(\mathbf{x}+\mathbf{s}) = \mathbf{F}(\mathbf{x}) + J(\mathbf{x})\mathbf{s} + \text{higher-order terms},$$

where $J(\mathbf{x})$ is the Jacobian matrix

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}.$$

Newton's method results when we replace $\mathbf{F}(\mathbf{x}+\mathbf{s}) = \mathbf{0}$ by $\mathbf{F}(\mathbf{x}) + J(\mathbf{x})\mathbf{s} = 0$. Solving for the vector $\mathbf{s}$ yields the iteration

$$\mathbf{x} \leftarrow \mathbf{x} - [J(\mathbf{x})]^{-1}\,\mathbf{F}(\mathbf{x}).$$

But what about the current approximation $\mathbf{v}^{2h}$? When we move to the coarse grid, it makes sense to restrict the current fine-grid approximation to the coarse grid, often using the same transfer operator used for the residual: $\mathbf{v}^{2h} = I_h^{2h}\mathbf{v}^h$.

Making these substitutions in the residual equation yields

$$A^{2h}(\underbrace{I_h^{2h}\mathbf{v}^h + \mathbf{e}^{2h}}_{\mathbf{u}^{2h}}) = A^{2h}(I_h^{2h}\mathbf{v}^h) + I_h^{2h}(\mathbf{f}^h - A^h(\mathbf{v}^h)). \tag{6.6}$$

The right side of this nonlinear system is known. The goal is to find or approximate the solution to this system, which we have denoted $\mathbf{u}^{2h}$. The coarse-grid error

approximation, $\mathbf{e}^{2h} = \mathbf{u}^{2h} - I_h^{2h}\mathbf{v}^h$, can then be interpolated up to the fine grid and used to correct the fine-grid approximation $\mathbf{v}^h$. This correction step takes the form

$$\mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h \mathbf{e}^{2h} \quad \text{or} \quad \mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h(\mathbf{u}^{2h} - I_h^{2h}\mathbf{v}^h).$$

The scheme we have just outlined is the most commonly used nonlinear version of multigrid. It is called the *full approximation scheme* (FAS) because the coarse-grid problem is solved for the full approximation $\mathbf{u}^{2h} = I_h^{2h}\mathbf{v}^h + \mathbf{e}^{2h}$ rather than the error $\mathbf{e}^{2h}$. A two-grid version of this scheme is described as follows:

Full Approximation Scheme (FAS)

- Restrict the current approximation and its fine-grid residual to the coarse grid: $\mathbf{r}^{2h} = I_h^{2h}(\mathbf{f}^h - A^h(\mathbf{v}^h))$ and $\mathbf{v}^{2h} = I_h^{2h}\mathbf{v}^h$.
- Solve the coarse-grid problem $A^{2h}(\mathbf{u}^{2h}) = A^{2h}(\mathbf{v}^{2h}) + \mathbf{r}^{2h}$.
- Compute the coarse-grid approximation to the error: $\mathbf{e}^{2h} = \mathbf{u}^{2h} - \mathbf{v}^{2h}$.
- Interpolate the error approximation up to the fine grid and correct the current fine-grid approximation: $\mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h \mathbf{e}^{2h}$.

There are several observations to be made about this method. It is worth noting that if A is a linear operator, then the FAS scheme reduces directly to the (linear) two-grid correction scheme (Exercise 1). Thus, FAS can be viewed as a generalization of the two-grid correction scheme to nonlinear problems.

Less obvious is the fact that an exact solution of the fine-grid problem is a fixed point of the FAS iteration (Exercise 2). This fixed point property, which is a desirable attribute of most iterative methods, means that the process stalls at the exact solution.

A third observation is that the FAS coarse-grid equation can be written as

$$A^{2h}(\mathbf{u}^{2h}) = \mathbf{f}^{2h} + \boldsymbol{\tau}_h^{2h},$$

where the *tau correction* $\boldsymbol{\tau}_h^{2h}$ is defined by

$$\boldsymbol{\tau}_h^{2h} = A^{2h}(I_h^{2h}\mathbf{v}^h) - I_h^{2h}A^h(\mathbf{v}^h).$$

One of the many consequences of this relationship is that the solution of the coarse-grid FAS equation, $\mathbf{u}^{2h}$, is not the same as the solution of the original coarse-grid equation $A^{2h}(\mathbf{u}^{2h}) = \mathbf{f}^{2h}$ because $\boldsymbol{\tau}_h^{2h} \neq \mathbf{0}$ generally. In fact, as FAS processing advances, $\mathbf{u}^{2h}$ begins to achieve *accuracy* that compares to that of the solution on the finest grid, albeit at the *resolution* of grid $2h$. This tau correction relationship allows us to view FAS as a way to alter the coarse-grid equations so that their approximation properties are substantially enhanced.

Here is another important observation. Because the second step in the above procedure involves a nonlinear problem itself, FAS involves an inner and an outer iteration; the outer iteration is the FAS correction scheme, while the inner iteration is usually a standard relaxation method such as nonlinear Gauss–Seidel. A true multilevel FAS process would be done recursively by approximating solutions to

the Ω^{2h} problem using the next coarsest grid, Ω^{4h}. Thus, FAS, like its linear counterparts, is usually implemented as a V-cycle or W-cycle scheme.

In earlier chapters, we saw the importance of full multigrid (FMG) for obtaining a good initial guess for the fine-grid problem. The convergence of nonlinear iterations depends even more critically on a good initial guess. Typically, the better the initial guess used on the fine grid, the more linear the fine-grid problem appears, and the more effective the fine-grid solver will be. When FMG is used for nonlinear problems, the interpolant $I_{2h}^h \mathbf{u}^{2h}$ is generally accurate enough to be in the basin of attraction of the fine-grid solver. Thus, whether we use Newton-multigrid or FAS V-cycles on each new level, we can expect one FMG cycle to provide accuracy to the level of discretization, unless perhaps the nonlinearity is exceptionally strong. These options are investigated in a numerical example later in the chapter.

Example: An algebraic problem. We need to warn the reader that this example should not be taken too seriously: it was devised only as a way to illustrate the *mechanics* of the FAS scheme. The problem does not really need the power of multigrid because it can be effectively treated by a good classical relaxation scheme. More importantly, as with most nonlinear problems, the example has subtleties that must be addressed before solution techniques should even be considered (Exercise 4).

Consider the following nonlinear algebraic system of n equations and n unknowns $\mathbf{u}^h \in \mathbf{R}^n$:

$$A_j^h(\mathbf{u}^h) \equiv u_j^h \, u_{j+1}^h = f_j^h, \qquad 1 \le j \le n\,, \tag{6.7}$$

where the vector $\mathbf{f}^h \in \mathbf{R}^n$ is given. (The notation $A_j^h(\mathbf{u}^h)$ means the jth component of $A^h(\mathbf{u}^h)$.) We impose the periodic boundary condition $u_{n+1}^h = u_1^h$ to close the problem. Suppose we have a fine-grid approximation, $\mathbf{v}^h$, obtained by relaxation. In passing, note that Gauss–Seidel relaxation is easy to formulate for this problem: it consists of the replacement steps

$$v_j^h \leftarrow \frac{f_j^h}{v_{j+1}^h}, \quad 1 \le j \le n. \tag{6.8}$$

To see what the FAS correction looks like, assume that n is a positive even integer and write coarse-grid residual equation (6.5) as

$$A_j^{2h}(\mathbf{v}^{2h} + \mathbf{e}^{2h}) - A_j^{2h}(\mathbf{v}^{2h}) = r_j^{2h}, \quad 1 \le j \le \frac{n}{2}.$$

In component form, these equations are

$$\begin{aligned} (v_j^{2h} + e_j^{2h})(v_{j+1}^{2h} + e_{j+1}^{2h}) - v_j^{2h} v_{j+1}^{2h} &= r_j^{2h} \qquad \text{or} \\ v_j^{2h} e_{j+1}^{2h} + v_{j+1}^{2h} e_j^{2h} + e_j^{2h} e_{j+1}^{2h} &= r_j^{2h}, \quad 1 \le j \le \frac{n}{2}. \end{aligned}$$

Notice how the term $v_j^{2h} v_{j+1}^{2h}$ cancels. Now, e_j^{2h} and e_{j+1}^{2h} are the unknowns in the jth equation; the coefficients v_j^{2h}, v_{j+1}^{2h}, and r_j^{2h} must come from the fine grid. We can obtain the terms v_j^{2h} and v_{j+1}^{2h} from the fine grid by injection:

$$v_j^{2h} = I_h^{2h} v_{2j}^h = v_{2j}^h, \quad v_{j+1}^{2h} = I_h^{2h} v_{2j+2}^h = v_{2j+2}^h.$$

Similarly, the coarse-grid residual (using injection) is given by

$$r_j^{2h} = I_h^{2h} r_{2j}^h = f_{2j}^h - v_{2j}^h v_{2j+1}^h.$$

The equations that must be solved on the coarse grid for e_j^{2h} are therefore given by

$$v_{2j}^h e_{j+1}^{2h} + v_{2j+2}^h e_j^{2h} + e_j^{2h} e_{j+1}^{2h} = f_{2j}^h - v_{2j}^h v_{2j+1}^h, \quad 1 \le j \le \frac{n}{2}.$$

This system corresponds to the general FAS equation (6.6). In this example, with a specific form for the operator A^h, we have simplified the right side of (6.6), resulting in an explicit system for e_j^{2h}. This system would typically be handled by a standard relaxation method. Note that Gauss–Seidel is again fairly simple because the jth equation is linear in e_j^{2h}, although the *system* of equations is nonlinear in the vector $\mathbf{e}^h$. In any case, after approximations to e_j^{2h} have been computed, they can then be interpolated up to the fine grid to correct $\mathbf{v}^h$. ◇◇

Example: Formulating FAS for a boundary value problem. Consider the two-point boundary value problem

$$\begin{aligned} -u''(x) + \gamma u(x) u'(x) &= f(x), \qquad 0 < x < 1, \\ u(0) = u(1) &= 0. \end{aligned} \tag{6.9}$$

The source term f and a constant $\gamma > 0$ are given. To set up the FAS solution to this problem, let Ω^h consist of the grid points $x_j = \frac{j}{n}$, for some positive even integer n, and let $u_j = u(x_j)$ and $f_j = f(x_j)$, for $j = 0, 1, \ldots, n$. Of the many possible ways to discretize this differential equation, consider the following finite difference scheme:

$$A_j^h(\mathbf{u}) = \frac{-u_{j-1}^h + 2u_j^h - u_{j+1}^h}{h^2} + \gamma u_j^h \left(\frac{u_{j+1}^h - u_{j-1}^h}{2h} \right) = f_j, \qquad 1 \le j \le n-1.$$

Note the use of a centered difference approximation to $u'(x_j)$. One consequence of this choice is that this *scalar* equation is linear in u_j^h even though the *vector* equation $A^h(\mathbf{u}^h) = \mathbf{f}^h$ is nonlinear in $\mathbf{u}^h$. Thus, nonlinear Gauss–Seidel requires no Newton step and can be stated explicitly as follows:

$$u_j^h \leftarrow 2 \frac{h^2 f_j + \left(u_{j-1}^h + u_{j+1}^h \right)}{4 + h\gamma(u_{j+1}^h - u_{j-1}^h)}, \qquad 1 \le j \le n-1\,. \tag{6.10}$$

To examine the FAS correction step, assume that an approximation $\mathbf{v}^h$ has been obtained on the fine grid. Residual equation (6.5), given by

$$A_j^{2h}(\mathbf{v}^{2h} + \mathbf{e}^{2h}) - A_j^{2h}(\mathbf{v}^{2h}) = r_j^{2h},$$

appears in component form as

$$\begin{aligned} &\frac{-(v_{j-1}^{2h} + e_{j-1}^{2h}) + 2(v_j^{2h} + e_j^{2h}) - (v_{j+1}^{2h} + e_{j+1}^{2h})}{4h^2} \\ &+ \ \gamma(v_j^{2h} + e_j^{2h}) \left(\frac{v_{j+1}^{2h} - v_{j-1}^{2h}}{4h} + \frac{e_{j+1}^{2h} - e_{j-1}^{2h}}{4h} \right) - \frac{-v_{j-1}^{2h} + 2v_j^{2h} - v_{j+1}^{2h}}{4h^2} \\ &- \ \gamma v_j^{2h} \left(\frac{v_{j+1}^{2h} - v_{j-1}^{2h}}{4h} \right) = r_j^{2h}, \quad 1 \le j \le \frac{n}{2} - 1\,. \end{aligned}$$

Canceling terms leaves the coarse-grid equation for the unknowns e_j^{2h}:

$$\frac{-e_{j-1}^{2h} + 2e_j^{2h} - e_{j+1}^{2h}}{4h^2} + \gamma v_j^{2h}\left(\frac{e_{j+1}^{2h} - e_{j-1}^{2h}}{4h}\right) + \gamma e_j^{2h}\left(\frac{v_{j+1}^{2h} - v_{j-1}^{2h}}{4h}\right)$$
$$+ \ \gamma e_j^{2h}\left(\frac{e_{j+1}^{2h} - e_{j-1}^{2h}}{4h}\right) = \underbrace{I_h^{2h}(f_j^h - A_j^h(v^h))}_{r_j^{2h}}.$$

As before, the terms $v_j^{2h}, v_{j+1}^{2h}, v_{j-1}^{2h}$, and r_j^{2h} are obtained by restriction from the fine grid.

This equation is the analogue of FAS equation (6.6), although we have written out the terms explicitly. Approximations to the solution, e_j^{2h}, of this equation must be computed using a relaxation method such as nonlinear Gauss–Seidel (which can again be carried out explicitly). These corrections are interpolated up to the fine grid and used to update the fine-grid approximation $\mathbf{v}^h$. ◇◇

Numerical example: FAS for a boundary value problem. To study the performance of FAS on boundary value problem (6.9), we choose the exact solution

$$u(x) = e^x(x - x^2),$$

which results in the source term

$$f(x) = (x^2 + 3x)e^x \ + \ \gamma\,(x^4 - 2x^2 + x)e^{2x}.$$

The problem is discretized on a grid with $n = 512$ (511 interior points). On the coarsest grid, consisting of one interior point and two boundary points, the problem is solved exactly by

$$u_1 = \frac{f_1}{8}.$$

In this experiment, we treat the problem using full weighting, linear interpolation, and a (2,1) FAS V-cycle based on nonlinear Gauss–Seidel (6.10). For $\gamma = 0$, the problem reduces to the one-dimensional Poisson equation, which is the standard model problem for multigrid. In this instance, the FAS V-cycle reduces to the standard V-cycle. As γ increases, the nonlinear term $\gamma u(x)u'(x)$ begins to dominate the problem.

Table 6.1 displays the performance of FAS for various values of γ. Starting with an initial guess of $\mathbf{v}^h = \mathbf{0}$, the FAS V-cycles are carried out until the norm of the residual vector is less than 10^{-10}. Displayed for each choice of γ is the number of FAS V-cycles required to achieve this tolerance, as well as the average convergence factor per cycle. For $\gamma \leq 10$, FAS performance is essentially the same as for the linear case, $\gamma = 0$. As γ increases, the performance degrades slowly until, for $\gamma \geq 40$, FAS no longer converges.

We see similar qualitative behavior if we change the exact solution to $u(x) = x - x^2$ so that $f(x) = 2 + \gamma\,(x - x^2)(1 - 2x)$. FAS performance for this problem is summarized in Table 6.2. Now the method converges for $\gamma < 100$, although performance degrades more noticeably as γ approaches $\gamma = 100$. For $\gamma \geq 100$, the method no longer converges. FAS is an effective solver for this problem, even for cases where the nonlinear term is relatively large.

	γ						
	0	1	10	25	35	39	≥ 40
Convergence factor	.096	.096	.096	.163	.200	.145	NC
Number of FAS cycles	11	11	11	14	16	13	NC

Table 6.1: *FAS performance for the problem* $-u'' + \gamma\, u\, u' = f$*, with the exact solution* $u(x) = e^x(x - x^2)$*, discretized on a grid with* 511 *interior points. Shown are average convergence factors and the number of FAS cycles needed to converge from a zero initial guess to a residual norm of* 10^{-10}*. NC indicates that the method did not converge.*

	γ						
	0	1	10	50	70	90	≥ 100
Convergence factor	.096	.096	.094	.148	.432	.728	NC
Number of FAS V-cycles	11	11	11	13	30	79	NC

Table 6.2: *FAS performance for the problem* $-u'' + \gamma\, u\, u' = f$*, with the exact solution* $u(x) = x - x^2$*, discretized on a grid with* 511 *interior points. Shown are average convergence factors and the number of FAC cycles needed to converge from a zero initial guess to a residual norm of* 10^{-10}*. NC indicates that the method did not converge.* ◇◇

Numerical example: Two-dimensional boundary value problem. We finish the numerical examples with a study of the performance of FAS and Newton solvers applied to the two-dimensional nonlinear problem

$$-\Delta u(x,y) + \gamma\, u(x,y)\, e^{u(x,y)} = f(x,y) \qquad \text{in } \Omega, \tag{6.11}$$

$$u(x,y) = 0 \qquad \text{on } \partial\Omega, \tag{6.12}$$

where Ω is the unit square $[0,1] \times [0,1]$. For $\gamma = 0$, this problem reduces to the model problem. We discretize this equation on uniform grids in both directions, with grid spacings of $h = \frac{1}{64}$ and $h = \frac{1}{128}$. Using the usual finite difference operator, the equation for the unknown $u_{i,j}$ at $(x_i, y_j) = (ih, jh)$ becomes

$$\frac{4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}}{h^2} + \gamma\, u_{i,j}\, e^{u_{i,j}} = f_{i,j}, \qquad 1 < i, j < n. \tag{6.13}$$

To enforce the boundary condition, we set

$$u_{0,j} = u_{N,j} = u_{i,0} = u_{i,N} = 0$$

wherever these terms appear in the equations.

We consider several approaches to solving this problem. An FAS solver can be implemented in a straightforward way. We use full weighting and linear interpolation for the intergrid transfer operators. For the coarse-grid versions of the nonlinear operator, we use discretization (6.13) with the appropriate grid spacing ($2h$, $4h$, ...). Because the individual component equations of the system are nonlinear, the nonlinear Gauss–Seidel iteration uses scalar Newton's method to solve

the (i,j)th equation for $u_{i,j}$ (Exercise 5):

$$u_{i,j} \leftarrow u_{i,j} - \frac{h^{-2}(4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}) + \gamma\, u_{i,j}\, e^{u_{i,j}} - f_{i,j}}{4h^{-2} + \gamma\,(1 + u_{i,j})\, e^{u_{i,j}}}. \tag{6.14}$$

The nonlinear Gauss–Seidel smoother is not intended to solve the system exactly, so we need not solve for $u_{i,j}$ exactly. Although no fixed rule exists to determine how many Newton steps are required in each nonlinear Gauss–Seidel sweep, a small number usually suffices (we used one Newton step in this example). It should be noted that the number of Newton steps used on the scalar problem can have a significant impact on both the effectiveness and the cost of the FAS algorithm.

In the previous example, it was simple to solve the equation on the coarsest grid exactly. This is not the case here, where, on a 3×3 grid with a single interior point, the equation to be solved for $u_{1,1}$ is

$$16u_{1,1} + \gamma\, u_{1,1}\, e^{u_{1,1}} = f_{1,1}.$$

Because this equation is nonlinear, the "solve" on the coarse grid requires the use of Newton's Method. All experiments presented here use a 3×3 grid as the coarsest grid, and in all cases it was determined experimentally that a single Newton step was sufficient there.

In this example, we use the source term

$$f(x,y) = 2((x - x^2) + (y - y^2)) \;+\; \gamma\,(x - x^2)(y - y^2)\, e^{(x-x^2)(y-y^2)},$$

which corresponds to the exact solution

$$u(x,y) = (x - x^2)(y - y^2).$$

The first test examines the effect of FAS for various choices of γ, from the linear case ($\gamma = 0$) through cases in which the nonlinear part of the operator dominates ($\gamma = 10,000$). For each test, the problem is solved on a 127×127 interior grid. The problem is deemed solved when the norm of the residual is less than 10^{-10}, at which point the average convergence rate is reported. Also reported is the number of V-cycles required to achieve the desired tolerance. The results appear in Table 6.3.

	γ					
	0	1	10	100	1000	10000
Convergence factor	.136	.135	.124	.098	.072	.039
Number of FAS cycles	12	12	11	11	10	8

Table 6.3: *FAS performance for the problem* $-\Delta u + \gamma\, u\, e^u = f$*, discretized on a grid with* 127×127 *interior points.*

One feature of interest is that this problem becomes *easier* to solve as the nonlinear term becomes more dominant. Note that this particular nonlinear term, ue^u, involves only the unknown $u_{i,j}$, and none of its neighboring values. Therefore, as the nonlinear term becomes more dominant, the problem becomes more local

in nature. As this occurs, the smoothing steps, being local solvers, become more effective. Indeed, with enough dominance of the nonlinear term, the problem could be solved entirely with the nonlinear Gauss–Seidel smoother. This phenomenon is analogous to increasing diagonal dominance in the linear case. Of course, this type of behavior would not be expected for other types of nonlinearity.

It is useful to examine how the FAS solver performs compared to Newton's method applied to the nonlinear system. The Jacobian matrix for this problem is a block tridiagonal system

$$J(\mathbf{u}) = \begin{bmatrix} J_1 & B & & & & & \\ B & J_2 & B & & & & \\ & B & J_3 & B & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & B & J_{N-2} & B & \\ & & & & B & J_{N-1} & \end{bmatrix},$$

where each of the block matrices is $(n-1)\times(n-1)$. The off-diagonal blocks B are all $-\frac{1}{h^2}$ times the identity matrix. The diagonal blocks are tridiagonal, with the constant value $-\frac{1}{h^2}$ on the super- and sub-diagonals. The diagonal entries of J_j, corresponding to the grid locations $(x_i, y_j) = (ih, jh)$ for fixed j and $1 \le i \le n-1$, are given by

$$(J_j)_{i,i} = \frac{4}{h^2} + \gamma\, u_{i,j}\, e^{u_{i,j}}.$$

To perform the Newton iteration, we choose two of the many ways to invert the Jacobian matrix. The first is to compute the LU decomposition of $J(\mathbf{u})$ and to use it to solve the system. With a lower triangular matrix L and an upper triangular matrix U such that $LU = J(\mathbf{u})$, we solve the system $J(\mathbf{u})\mathbf{s} = F(\mathbf{u})$, first solving $L\mathbf{y} = F(\mathbf{u})$ and then solving $U\mathbf{s} = \mathbf{y}$. Because $J(\mathbf{u})$ is sparse and has a narrow band of nonzero coefficients, the factors L and U are also sparse, narrow-banded matrices. Indeed, the banded LU decomposition can be computed quite efficiently; and because the factors L and U are triangular, solving for $\mathbf{y}$ and $\mathbf{s}$ is also fast. Table 6.4 gives the results of applying Newton's method, with the band LU solver, to the problem $-\Delta u + \gamma\, u\, e^u = f$, using a 63×63 interior grid.

	γ					
	0	1	10	100	1000	10000
Convergence factor	2.6e-13	3.9e-5	7.4e-5	3.2e-4	1.9e-4	1.2e-4
Number of Newton iterations	1	3	3	4	4	4

Table 6.4: *Performance of Newton's Method for the problem* $-\Delta u + \gamma\, u\, e^u = f$, *discretized on a* 63×63 *interior grid.*

A comparison of Tables 6.3 and 6.4 indicates that convergence of Newton's method is much faster than that of FAS. It is well known that Newton's method converges quadratically, so this is not surprising. To make this comparison useful, however, it is important to compare the computational costs of the methods. We present some empirical evidence on this question shortly.

First, however, we consider an alternative method for inverting the Jacobian system in the Newton iteration, namely, V-cycles. Certainly in the linear case

($\gamma = 0$), we know this is an effective method: both the original and the Jacobian problems are discrete Poisson equations. As γ increases, the diagonal dominance of the Jacobian improves, so we should expect even better performance.

If we solve the Jacobian system to the same level of accuracy by V-cycles as was done with the LU decomposition, the results should be essentially the same as in Table 6.4. However, it is probably much more efficient to use only a few V-cycles to solve the Jacobian system approximately. This technique is known as the *inexact Newton's method* [8], which we denote Newton-MG.

We now compare the performance of FAS, Newton's method, and Newton-MG in the following numerical experiment. Consider the same operator as before with a different source term:

$$-\Delta u + \gamma u e^u = \left(\left(9\pi^2 + \gamma e^{(x^2-x^3)\sin(3\pi y)}\right)(x^2 - x^3) + 6x - 2\right)\sin(3\pi y). \quad (6.15)$$

With $\gamma = 10$, the exact solution is $u(x, y) = (x^2 - x^3)\sin(3\pi y)$. This solution (as opposed to the previous polynomial solution) results in a nontrivial discretization error. The problem is discretized with $n = 128$ so that the interior grid is 127×127.

Method	No. outer iterations	No. inner iterations	Megaflops
Newton	3	–	1660.6
Newton-MG	3	20	56.4
Newton-MG	4	10	38.5
Newton-MG	5	5	25.1
Newton-MG	10	2	22.3
Newton-MG	19	1	24.6
FAS	11	–	27.1

Table 6.5: *Comparison of FAS, Newton, and Newton-multigrid methods for the problem* $-\Delta u + \gamma\, u\, e^u = f$ *on a* 127×127 *grid. In all cases, a zero initial guess is used.*

Table 6.5 shows the costs of FAS, Newton (with a direct solve), and Newton-MG applied to (6.15), resulting in the nonlinear system (6.13). The iteration is stopped when the residual norm is less than 10^{-10}. The column labeled *outer iterations* lists how many V-cycles (for FAS) or Newton steps (for Newton and Newton-MG) are required to achieve the desired tolerance. For Newton-MG, we varied the number of V-cycles used to solve the Jacobian system. The column labeled *inner iterations* gives the number of V-cycles used in the approximate inversion of the Jacobian system. The last column, labeled *Megaflops*, is the number of millions of floating-point operations required to achieve the desired tolerance (as determined by the MATLAB *flops* function). For this example, using these performance measurements, it appears that both the multigrid-based methods are much more efficient than Newton's method using a direct solver. Furthermore, Newton-MG compares well to FAS when the number of inner MG iterations is properly tuned.

These results should not be taken too seriously. While the convergence properties may be representative for a fairly broad class of nonlinear problems, the operation counts are likely to vary dramatically with the character of the nonlinearity, details of implementation, computer architecture, and programming language. It should also be remembered that we have not accounted for the cost of evaluating the

nonlinear function, which is highly problem-dependent. Therefore, it may be risky to draw any general conclusions from this single experiment. On the other hand, it does seem fairly clear that the multigrid-based methods will outperform Newton's method using direct solvers when the problems are large. It is also clear that only a few V-cycles should be used in the Newton scheme if it is to be competitive with FAS. Finally, in the end, there may be very little difference in the performance between the two carefully designed multigrid-based schemes. The choice may depend largely on convenience and other available algorithm features (for example, τ-extrapolation in FAS [4]).

There is, of course, one further option we should consider: combining the Newton and FAS methods with an FMG scheme. The idea is to apply the basic solver (either Newton or FAS) on successively finer grids of the FMG cycle: for each new fine grid, the initial guess is obtained by first solving the nonlinear problem on the next coarser grid. In the linear case, we showed that convergence to the level of discretization was achieved in one FMG cycle. Table 6.6 shows the results of applying the FMG–FAS combination to (6.15); it suggests that we can hope for the same kind of performance in the nonlinear case.

The discrete L^2 norms are shown for the residual and error (difference between computed and sampled continuous solutions), after one FMG–FAS cycle and eight subsequent FAS V-cycles on the fine grid. Both the FMG cycle and the FAS V-cycles were performed using nonlinear Gauss–Seidel (2,1) sweeps. Observe that the norm of the error is reduced to 2.0×10^{-5} by the FMG–FAS cycle alone. Further FAS V-cycling does not reduce the error, indicating that it has reached the level of discretization error. However, subsequent FAS V-cycles do reduce the residual norm further until it reaches the prescribed tolerance of 10^{-10}. The column labeled *Mflops* in the table gives the cumulative number of floating-point operations after each stage of the computation.

Cycle	$\|\|\mathbf{r}^h\|\|_h$	Ratio	$\|\|\mathbf{e}\|\|_h$	Mflops
FMG–FAS	1.07e−2		2.00e−5	3.1
FAS V 1	6.81e−4	0.064	2.44e−5	5.4
FAS V 2	5.03e−5	0.074	2.49e−5	7.6
FAS V 3	3.89e−6	0.077	2.49e−5	9.9
FAS V 4	3.25e−7	0.083	2.49e−5	12.2
FAS V 5	2.98e−8	0.092	2.49e−5	14.4
FAS V 6	2.94e−9	0.099	2.49e−5	16.7
FAS V 7	3.01e−10	0.102	2.49e−5	18.9
FAS V 8	3.16e−11	0.105	2.49e−5	21.2

Table 6.6: *Performance of the FMG–FAS cycle, followed by eight FAS V-cycles, on* $-\Delta u + \gamma u e^u = f$*, with* $\gamma = 10$*. The grid size is* 127×127*. Note that one FMG–FAS cycle reduces the error to the level of discretization error, and that subsequent FAS V-cycles further reduce the residual norm quickly to the prescribed tolerance of* 10^{-10}*.*

We show analogous results in Table 6.7 for FMG with a Newton solver applied to (6.15). Here we again use FMG, applying one step of Newton-MG on each level in the FMG process. Each Newton step starts with an initial guess from the next coarser grid and uses one (2,1) V-cycle. The discrete L^2 norms are shown for the residual and error after one FMG–Newton-MG cycle followed by subsequent

Cycle	$\|\|\mathbf{r}^h\|\|_h$	Ratio	$\|\|\mathbf{e}\|\|_h$	Mflops
FMG–Newton-MG	1.06e−002		2.50e−005	2.4
Newton-MG 1	6.72e−004	0.063	2.49e−005	4.1
Newton-MG 2	5.12e−005	0.076	2.49e−005	5.8
Newton-MG 3	6.30e−006	0.123	2.49e−005	7.5
Newton-MG 4	1.68e−006	0.267	2.49e−005	9.2
Newton-MG 5	5.30e−007	0.315	2.49e−005	10.9
Newton-MG 6	1.69e−007	0.319	2.49e−005	12.6
Newton-MG 7	5.39e−008	0.319	2.49e−005	14.3
Newton-MG 8	1.72e−008	0.319	2.49e−005	16.0
Newton-MG 9	5.50e−009	0.319	2.49e−005	17.7
Newton-MG 10	1.76e−009	0.319	2.49e−005	19.4
Newton-MG 11	5.61e−010	0.319	2.49e−005	21.1
Newton-MG 12	1.79e−010	0.319	2.49e−005	22.8
Newton-MG 13	5.71e−011	0.319	2.49e−005	24.5

Table 6.7: *Performance of the FMG–Newton-MG cycle, followed by* 13 *Newton-MG steps, on* $-\Delta u + \gamma u e^u = f$*, with* $\gamma = 10$*. The grid size is* 127×127*. Note that one FMG–Newton-MG cycle reduces the error to the level of discretization error, and that subsequent Newton-MG steps on the fine grid further reduce the residual error to the prescribed tolerance of* 10^{-10}*.*

Newton-MG cycles on the fine grid. The results are very similar to those for FMG–FAS in Table 6.6. Observe that the norm of the actual error is reduced to the level of discretization error by one FMG–Newton cycle. Subsequent Newton-MG cycles do, however, continue to reduce the discrete L^2 norm of the residual effectively to below the prescribed tolerance.

Both of these methods reduce the error to the level of discretization in one FMG cycle. The flop count indicates that the methods are similar in cost, with the FMG–Newton-MG cycle somewhat less expensive (2.4 Mflops) than the FMG–FAS cycle (3.1 Mflops). However, the individual Newton-MG steps on the fine grid, although cheaper, are not quite as effective as FAS V-cycles for reducing the residual norm. Indeed, if the goal is to reduce the residual norm to 10^{-10}, it is somewhat less expensive (21.2 vs. 26.2 Mflops) to use FAS than Newton-MG. We remind the reader, however, that these flop counts, like those reported earlier, should not be taken too seriously: comparative measures of efficiency depend critically on specific implementation details, computing environment, and problem characteristics. The major conclusion to be reached is that both methods are very efficient and robust.

One of the goals of FMG is to obtain (efficiently) results that are comparable to the accuracy of the discretization. For nonlinear problems, the nested iteration feature of FMG improves the likelihood that initial guesses will lie in the basin of attraction of the chosen iterative method, which accounts for some of its effectiveness. We do not claim that FMG solves all the difficulties that nonlinear problems present. Rather, FMG should be considered a potentially powerful tool that can be used to treat nonlinear problems that arise in practice. ◇◇

So far we have developed FAS in a fairly mechanical way. We showed how the development of multigrid for linear problems can be mimicked to produce a scheme that makes sense for nonlinear equations. However, what we have not done

is motivate this development from basic principles in a way that suggests why it works. We attempt to do so now.

Here is the central question in treating nonlinearities by multigrid: When working on the fine grid with the original equation in the form $A^h(\mathbf{v}^h + \mathbf{e}^h) = \mathbf{f}^h$, how can the error for this equation be effectively represented on a coarser grid?

This is analogous to asking: How do you move from the *continuous problem* to the fine grid; that is, how do you effectively discretize the differential equation $A(v + e) = f$ with a known function v? For one possible answer, we turn to the second example above ($-u'' + \gamma uu' = f$) with u replaced by $v + e$:

$$-(v + e)'' + \gamma(v + e)(v + e)' = f.$$

It is important first to think of e and e' as small quantities so that the dominant part of this equation is the term $-v'' + \gamma vv'$, which must be treated carefully. To expose this term in the equation, we expand the products to obtain

$$-(v'' + e'') + \gamma(vv' + v'e + ve' + ee') = f.$$

Since $-v'' + \gamma vv'$ is known, we simply move it to the right side to obtain the *differential residual equation*

$$\gamma(ve' + v'e + ee') = f - (-v'' + \gamma vv'). \tag{6.16}$$

A natural way to discretize this equation begins by evaluating the right side,

$$r = f - (-v'' + \gamma vv') = f - A(v),$$

at the grid points; this produces a vector $r_j^h = r(x_j)$. This way of transferring r from the differential setting to Ω^h is analogous to transferring the fine-grid residual $\mathbf{r}^h$ to Ω^{2h} by injection: $r_j^{2h} = r_{2j}^h$.

For the left side of residual equation (6.16), we need a sensible scheme for evaluating the coefficients v and v' on Ω^h. For v, the natural choice is again to evaluate it at the grid points: $v_j^h = v(x_j)$. For v', we can use a central difference approximation at the grid points:

$$(v')_j^h \approx \frac{v(x_{j+1}) - v(x_{j-1})}{2h}.$$

We are now left with the task of representing e and e' on Ω^h. This is naturally done using the expressions e_j^h and

$$(e')_j^h \approx \frac{e_{j+1}^h - e_{j-1}^h}{2h},$$

respectively.

Putting all of this together gives us a fine-grid approximation to the differential residual equation. We write it at grid point x_{2j} of the fine grid as follows:

$$\gamma v_{2j}^h \left(\frac{e_{2j+1}^h - e_{2j-1}^h}{2h} \right) + \gamma e_{2j}^h \left(\frac{v_{2j+1}^h - v_{2j-1}^h}{2h} \right) + \gamma e_{2j}^h \left(\frac{e_{2j+1}^h - e_{2j-1}^h}{2h} \right)$$
$$= \underbrace{f_{2j}^h - A_{2j}^h(v^h)}_{r_{2j}^h}.$$

We have shown how to move from the continuum to the fine grid in a natural way. How do we make the analogous move from the fine grid to the coarse grid? Terms evaluated at x_{2j} can be restricted directly to the coarse grid; for example, $v_{2j}^h = v_j^{2h}$. The difference approximations can be written as analogous centered differences with respect to the coarse-grid points. Making these replacements, we have the coarse-grid approximation to the fine-grid residual equation:

$$\gamma v_j^{2h}\left(\frac{e_{j+1}^{2h}-e_{j-1}^{2h}}{4h}\right)+\gamma e_j^{2h}\left(\frac{v_{j+1}^{2h}-v_{j-1}^{2h}}{4h}\right)+\gamma e_j^{2h}\left(\frac{e_{j+1}^{2h}-e_{j-1}^{2h}}{4h}\right)=(I_h^{2h}r^h)_j.$$

This is precisely the FAS equation that we derived in the example above.

We just developed FAS as a way to go from the fine to the coarse grid and showed that it can be viewed as a natural extension of the discretization process that goes from the continuum to the fine grid. This exposes a general multigrid coarsening principle: *What is good for discretization is probably good for coarsening.* There are notable exceptions to this principle, but the discretization process is always a good place to start for guidance in constructing the coarse-grid correction scheme.

Along these lines, notice that, in moving from the continuum to the fine grid, we could have computed the approximation to v' by reversing the order of the steps, that is, by *first* computing the derivative v', *then* evaluating it at the grid points: $(v')_i^h \approx v'(x_i)$. When moving from the fine grid to the coarse grid, this choice means we use the fine-grid difference approximation as a coefficient in the coarse-grid equation. Its coarse-grid correction formula is

$$\gamma v_j^{2h}\left(\frac{e_{j+1}^{2h}-e_{j-1}^{2h}}{4h}\right)+\gamma e_j^{2h}\left(\frac{v_{2j+1}^{h}-v_{2j-1}^{h}}{2h}\right)+\gamma e_j^{2h}\left(\frac{e_{j+1}^{2h}-e_{j-1}^{2h}}{4h}\right)=(I_h^{2h}r^h)_j.$$

Note that the only difference over FAS is in the coefficient of the second term.

This alternative illustrates that FAS is not necessarily the most direct way to treat the nonlinearity. In many cases, there are coarsening strategies that are more compatible with the discretization processes. See [15] for examples that use discretizations based on projections, as the finite element method does. On the other hand, FAS is advantageous as a general coarsening approach because of its very convenient form

$$A^{2h}(\mathbf{v}^{2h}+\mathbf{e}^{2h})-A^{2h}(\mathbf{v}^{2h})=\mathbf{r}^{2h}\,.$$

Exercises

1. **Linear problems.** Assume that A^h is actually a linear operator: $A^h(\mathbf{u}^h) = A^h\mathbf{u}^h$. Show that formula (6.5) for the FAS coarse-grid correction reduces to the formula for the two-grid correction scheme given in Chapter 3. Do this first by showing that the coarse-grid equations are the same, then by showing that the corrections to the fine-grid approximations are the same.

2. **Fixed point property.** Assume that relaxation has the fixed point property, namely, that if the exact solution $\mathbf{u}^h$ of equation (6.4) is used as the initial guess $\mathbf{v}^h$, then the result is $\mathbf{u}^h$. Assume also that the coarse-grid equation has a unique solution.

(a) Assume that the coarse-grid equation is solved exactly. Show that FAS also has this fixed point property.

(b) Now show that this fixed point property holds even if the exact solver of the coarse-grid equations is replaced by an appropriate approximate solver. Assume only that the coarse-grid solver itself exhibits this fixed point property.

3. **Residual equation.** You may have wondered why the FAS scheme is based on the residual equation and not the original equation $A^h(\mathbf{u}^h) = \mathbf{f}^h$. To see this more clearly, consider the following coarse-grid approximation to $A^h(\mathbf{v}^h + \mathbf{e}^h) = \mathbf{f}^h$:

$$A^{2h}(\mathbf{v}^{2h} + \mathbf{e}^{2h}) = I_h^{2h}\mathbf{f}^h\,.$$

If this is used in place of the coarse-grid equation $A^{2h}(\mathbf{v}^{2h}+\mathbf{e}^{2h}) = A^{2h}(\mathbf{v}^{2h})+\mathbf{r}^{2h}$, show that FAS generally loses both fixed point properties described in the previous exercise. Thus, for example, if you start with the exact fine grid solution, you will not get it back after the first iteration. Note also that the solution of this coarse-grid equation does not change from one FAS cycle to the next, so that it cannot properly represent smooth components of the fine-grid error, *which do change*.

4. **Example revisited.** Here we show that problem (6.7) of the first example is subtle in the sense that it has a solution if and only if the f_j satisfy a certain compatibility condition. For this purpose, assume that $f_j \neq 0$ for every j.

(a) Remembering that n is an even positive integer, show that if $\mathbf{u}$ satisfies (6.7), then

$$u_1 = \left(\frac{f_1 f_3 \cdots f_{n-1}}{f_2 f_4 \cdots f_n}\right) u_1\,.$$

(b) Now show that u_1 must be nonzero and that we therefore have the compatibility condition

$$f_1 f_3 \cdots f_{n-1} = f_2 f_4 \cdots f_n\,.$$

(c) Assuming that this compatibility condition is satisfied, show that the solution to (6.7) is determined up to an arbitrary multiplicative constant.

5. **Scalar Newton's methods within nonlinear Gauss–Seidel.** Show that in solving the component equations of (6.11) for $u_{i,j}$, scalar Newton's method takes the form given in (6.14).

6. **Formulating methods of solution.** Apply parts (a)–(e) to the following boundary value problems:

$$\text{(i)}\quad \begin{aligned} &u''(x) + e^u = 0,\\ &u(0) = u(1) = 0. \end{aligned} \qquad \text{(ii)}\quad \begin{aligned} &u''(x) + u^2 = 1,\\ &u(0) = u(1) = 0. \end{aligned}$$

(a) Use finite difference to discretize the problem. Write out a typical equation of the resulting system of equations.

(b) Determine if nonlinear Gauss–Seidel can be formulated explicitly. If so, write out a typical update step. If not, formulate a typical Newton step that must be used.

(c) Formulate Newton's method for the system and identify the linear system that needs to be solved at each step.

(d) Formulate the solution by FAS, indicating clearly the systems that must be solved on Ω^h and Ω^{2h} .