Chapter 14

The Method of Steepest Descent and the Conjugate Gradient Method

In this chapter we consider a class of methods for solving linear systems of equations when the matrix of coefficients is both symmetric and positive definite. (See Appendix A for the definitions of these terms.) Although we are interested primarily in the application of these methods to the solution of difference schemes for elliptic equations, these methods can be applied to any symmetric and positive definite system. We begin by discussing the method of steepest descent and then the conjugate gradient method, which can be regarded as an acceleration of the method of steepest descent. Our approach to the conjugate gradient method is based on that of Concus, Golub, and O'Leary in [10]. There have been many variations and extensions of the conjugate gradient method that cannot be discussed here. A good reference for these additional topics is the book by Hageman and Young [28] and the compendium of iterative methods by Barrett et al. [4].

14.1 The Method of Steepest Descent

We consider a system of linear equations

$$Ax = b, \tag{14.1.1}$$

where matrix A is symmetric and positive definite. As in the previous chapter we will let K be the order of the matrix. Consider also the function $F(y)$ defined by

$$F(y) = \tfrac{1}{2}\,(y - x, A(y - x))\,, \tag{14.1.2}$$

where x is the solution to (14.1.1) and $(\cdot, \cdot)$ is the usual inner product on R^K. Obviously, the function F has a unique minimum at y equal to x, the solution of (14.1.1). Similarly, the function E given by

$$E(y) = F(y) - F(0) = \tfrac{1}{2}(y, Ay) - (y, b) \tag{14.1.3}$$

has a unique minimum at the solution of (14.1.1). Both the method of steepest descent and the conjugate gradient method are iterative methods that reduce the value of E at each step until a vector y is obtained for which $E(y)$ is minimal or nearly minimal. In many

applications the function $E(y)$ represents a quantity of significance, such as the energy of the system. In these cases the solution of (14.1.1) is the state of minimum energy.

We first consider the method of steepest descent. The gradient of the function $E(y)$ is the vector

$$G(y) = Ay - b = -r, \tag{14.1.4}$$

where r is called the residual (see Exercise 14.1.1). Since the gradient of a function points in the direction of steepest ascent, to decrease the value of the function it is advantageous to go in the direction opposite of the gradient, which is the direction of steepest descent. The method of steepest descent, starting from an initial vector x^0, is given by

$$x^{k+1} = x^k + \alpha_k r^k, \tag{14.1.5}$$

where

$$r^k = b - Ax^k$$

and α_k is some parameter.

The notation we will use is that lowercase Roman letters will denote vectors and have superscripts, and Greek letters will denote scalar quantities and have subscripts. The norm of a vector v will be denoted by $|v|$, where $|v| = (v, v)^{1/2}$.

The parameter α_k in (14.1.5) will be chosen so that $E(x^{k+1})$ is minimal. We have

$$\begin{aligned} E\left(x^{k+1}\right) &= E\left(x^k + \alpha_k r^k\right) \\ &= \tfrac{1}{2}\left(x^k, Ax^k\right) + \alpha_k\left(r^k, Ax^k\right) + \tfrac{1}{2}\alpha_k^2\left(r^k, Ar^k\right) - \left(x^k, b\right) - \alpha_k\left(r^k, b\right) \\ &= E\left(x^k\right) - \alpha_k\left(r^k, r^k\right) + \tfrac{1}{2}\alpha_k^2\left(r^k, Ar^k\right). \end{aligned}$$

This expression is a quadratic function in α_k and has a minimum for some value of α_k. We find the minimum as follows. Setting $\partial E/\partial\alpha_k = 0$, we find that α_k given by

$$\alpha_k = \frac{\left(r^k, r^k\right)}{\left(r^k, Ar^k\right)} = \frac{|r^k|^2}{\left(r^k, Ar^k\right)} \tag{14.1.6}$$

is the value at which $E(x^{k+1})$ is minimal.

We now derive some consequences of this choice of α_k. From (14.1.5) we have that

$$r^{k+1} = r^k - \alpha_k Ar^k$$

and so from (14.1.6),

$$\left(r^{k+1}, r^k\right) = \left(r^k, r^k\right) - \alpha_k\left(r^k, Ar^k\right) = 0, \tag{14.1.7}$$

showing that consecutive residuals are orthogonal. For this optimal choice of α_k we have

$$E\left(x^{k+1}\right) = E\left(x^k\right) - \frac{1}{2}\frac{|r^k|^4}{\left(r^k, Ar^k\right)}, \tag{14.1.8}$$

showing that $E(x^k)$ will decrease as k increases until the residual is zero. Notice also from the definitions of $E\left(x^k\right)$ and r^k that

$$E\left(x^k\right) = \tfrac{1}{2}\left(A^{-1}r^k, r^k\right) - F(0)$$

and hence (14.1.8) is equivalent to

$$\left(A^{-1}r^{k+1},\ r^{k+1}\right) = \left(A^{-1}r^k,\ r^k\right) - \frac{|r^k|^4}{(r^k,\ Ar^k)}.$$

We now collect the formulas for steepest descent:

$$x^{k+1} = x^k + \alpha_k r^k, \tag{14.1.9a}$$

$$r^{k+1} = r^k - \alpha_k A r^k, \tag{14.1.9b}$$

$$\alpha_k = \frac{|r^k|^2}{\left(r^k,\, Ar^k\right)}. \tag{14.1.9c}$$

Notice that to implement the method, we need only one matrix multiplication per step; also, there is no necessity for storing the matrix A. Often A is quite sparse, as in solving linear systems arising from elliptic systems of finite difference equations, and we need only a means to generate the vector Ar given the vector r. Formula (14.1.9b) should be used instead of the formula $r^k = b - Ax^k$ to compute the residual vectors r^k. When using the finite precision of a computer, there is a loss of significant digits when the residual is computed as $b - Ax^k$, since the two vectors b and Ax^k will be nearly the same when k is not too small. The formula (14.1.9b) avoids this problem.

Although our derivation of the steepest descent method relied on matrix A being both symmetric and positive definite, we can apply the algorithm (14.1.9) in case A is not symmetric. The following theorem gives conditions on which the method will converge.

Theorem 14.1.1. *If A is a positive definite matrix for which A^TA^{-1} is also positive definite, then the algorithm given by (14.1.9) converges to the unique solution of (14.1.1) for any initial iterate x^0.*

Proof. First note that if A is positive definite, then A^{-1} is also positive definite, and if A^TA^{-1} is positive definite, we have that there are constants c_0 and c_1 such that

$$c_0(x, A^{-1}x) \le (x, A^TA^{-1}x) \tag{14.1.10}$$

and

$$c_1(x, Ax) \le (x, x) \tag{14.1.11}$$

for all vectors x (see Exercise 14.1.2). We now consider $\left(r^{k+1}, A^{-1}r^{k+1}\right)$, where $r^0 = b - Ax^0$ and r^{k+1} depends on r^k by (14.1.9b). We have

$$\begin{aligned}\left(r^{k+1}, A^{-1}r^{k+1}\right) &= \left(r^k, A^{-1}r^k\right) - \alpha_k\left(r^k, r^k\right) - \alpha_k\left(Ar^k, A^{-1}r^k\right) + \alpha_k^2\left(r^k, Ar^k\right)\\ &= \left(r^k, A^{-1}r^k\right) - \alpha_k\left(r^k, A^TA^{-1}r^k\right)\end{aligned}$$

by the definition of α_k. Now using (14.1.11) we have

$$\alpha_k = \frac{(r^k, r^k)}{(r^k, Ar^k)} \geq c_1$$

and thus, by (14.1.10),

$$\left(r^{k+1}, A^{-1}r^{k+1}\right) \leq \left(r^k, A^{-1}r^k\right)(1 - c_0c_1) \quad \text{for } k \geq 0. \tag{14.1.12}$$

Notice that $1 - c_0c_1$ is nonnegative, since A^{-1} is positive definite. Therefore,

$$\left(r^k, A^{-1}r^k\right) \leq \left(r^0, A^{-1}r^0\right)(1 - c_0c_1)^k$$

and thus $\left(r^k, A^{-1}r^k\right)$ tends to zero.

But r^k, given by (14.1.9b), is $b - Ax^k$, as can be shown by induction. Since A^{-1} is positive definite, we have that the vectors r^k converge to zero, and because

$$x^k = A^{-1}(b - r^k)$$

it follows that the vectors x^k converge to $A^{-1}b$, which is the unique solution of (14.1.1). □

Corollary 14.1.2. *If A is symmetric and positive definite, then the steepest descent method converges.*

The estimate (14.1.12) shows that if the product c_0c_1 can be taken to be close to 1, then the method of steepest descent will converge quite rapidly. As can be seen from Exercise 14.1.2, one way of having c_0c_1 close to 1 is if A is close to being a multiple of the identity matrix. However, steepest descent can often be quite slow, and this usually occurs because the residuals oscillate. That is, in spite of (14.1.7), we can have r^{k+2} be in essentially the same direction as r^k or $-r^k$.

Because the method of steepest descent is often quite slow, we consider several means to accelerate it. One method that accelerates steepest descent is the conjugate gradient method, which is the subject of the next several sections.

Exercises

14.1.1. Using the relation

$$E(y + z) = E(y) + (z, Ay - b) + O(|z|^2)$$

for the function $E(y)$ given by (14.1.3), verify that the gradient of the function $E(y)$ is $G(y) = Ay - b$, as asserted in (14.1.4).

14.1.2. Show that the constants c_0 and c_1 of (14.1.10) and (14.1.11) can be taken to be

$$c_0 = \frac{\lambda_3}{\lambda_2} \quad \text{and} \quad c_1 = \frac{1}{\lambda_1},$$

where λ_1 and λ_2 are the greatest eigenvalues and λ_3 is the least eigenvalue of

$$\tfrac{1}{2}(A + A^T), \qquad \tfrac{1}{2}(A^{-1} + A^{-T}), \quad \text{and} \quad \tfrac{1}{2}(A^{-1}A^T + AA^{-T}),$$

respectively.

14.1.3. Consider the matrix

$$A = \begin{pmatrix} 1 & b \\ 0 & 2 \end{pmatrix}.$$

(a) Show that A is positive definite if $|b| < 2\sqrt{2}$.

(b) Show that $A^T A^{-1}$ is positive definite if $|b| < 4/3$.

14.1.4. Show that if $|b| < 2$, then $|r^{k+2}| < |r^k|$ when the steepest descent algorithm is applied to the matrix of Exercise 14.1.3. Conclude that the method converges when $|b| < 2$. *Hint:* Show that if $r^k = \begin{pmatrix} x \\ y \end{pmatrix}$, then

$$r^{k+1} = \begin{pmatrix} y \\ -x \end{pmatrix} \frac{x(x - by)}{(x^2 + bxy + 2y^2)}.$$

14.1.5. Discuss the relationship between the example of Exercise 14.1.4 and Theorem 14.1.1 when $4/3 \le |b| < 2$.

14.1.6. Show that the method of steepest descent applied to the matrix of Exercise 14.1.3 does not converge for $|b| \ge 2$. *Hint:* Consider $r^0 = (\alpha, 1)^T$, where $\alpha^2 + \alpha - 1 = 0$.

14.1.7. Prove the Cauchy–Schwarz inequality for a symmetric, positive definite matrix A:

$$(x, Ay) \le (x, Ax)^{1/2}(y, Ay)^{1/2}.$$

Hint: Consider $(\alpha x - \beta y, A(\alpha x - \beta y))$.

14.2 The Conjugate Gradient Method

The conjugate gradient method can be viewed as an acceleration of steepest descent. We begin our derivation of the method by writing

$$x^{k+1} = x^k + \alpha_k \left[r^k + \gamma_k (x^k - x^{k-1}) \right]$$

for some scalar parameters α_k and γ_k. This formula shows that the new change in position, $x^{k+1} - x^k$, is a linear combination of the steepest descent direction and the previous change in position $x^k - x^{k-1}$. We rewrite the preceding formula as

$$x^{k+1} = x^k + \alpha_k p^k,$$

where

$$\begin{aligned} p^k &= r^k + \gamma_k \left(x^k - x^{k-1}\right) = r^k + \gamma_k \alpha_{k-1} p^{k-1} \\ &= r^k + \beta_{k-1} p^{k-1}. \end{aligned}$$

Combining these formulas we have

$$x^{k+1} = x^k + \alpha_k p^k, \tag{14.2.1a}$$
$$r^{k+1} = r^k - \alpha_k A p^k, \tag{14.2.1b}$$
$$p^{k+1} = r^{k+1} + \beta_k p^k, \tag{14.2.1c}$$

where the parameters α_k and β_k are to be determined. The vector p^k is called the *search direction* to the kth iteration.

We now wish to determine the parameters α_k and β_k and also determine what p^0 should be so that (14.2.1) converges as rapidly as possible. As with steepest descent, we wish to choose x^{k+1} so that $E\left(x^{k+1}\right)$ is minimal. To begin we assume that p^k is known, and we choose α_k so that $E\left(x^{k+1}\right)$ is minimized. We have

$$\begin{aligned} E\left(x^{k+1}\right) &= \tfrac{1}{2}\left(x^k, Ax^k\right) + \alpha_k\left(p^k, Ax^k\right) + \tfrac{1}{2}\alpha_k^2\left(p^k, Ap^k\right) - \left(x^k, b\right) - \alpha_k\left(p^k, b\right) \\ &= E\left(x^k\right) - \alpha_k\left(p^k, r^k\right) + \tfrac{1}{2}\alpha_k^2\left(p^k, Ap^k\right). \end{aligned}$$

By considering the derivative of $E(x^{k+1})$ with respect to α_k, we obtain that

$$\alpha_k = \frac{\left(p^k, r^k\right)}{\left(p^k, Ap^k\right)} \quad \text{for } k \geq 0 \tag{14.2.2}$$

is the optimal value of α_k. Using this value of α_k we have

$$E\left(x^{k+1}\right) = E\left(x^k\right) - \frac{1}{2}\frac{\left(p^k, r^k\right)^2}{\left(p^k, Ap^k\right)}.$$

We first consider the case $k = 0$, where we have complete freedom to choose p^0. From this formula we see that r^0 is a good choice for p^0, since it will make $E\left(x^1\right)$ less $E\left(x^0\right)$. From now on we will assume $p^0 = r^0$; later on we will see other advantages to this choice. Next we use (14.2.2) with (14.2.1b) to give

$$\left(p^k, r^{k+1}\right) = \left(p^k, r^k\right) - \alpha_k\left(p^k, Ap^k\right) = 0.$$

Then using this relation with (14.2.1c), we have

$$\begin{aligned}\left(p^{k+1}, r^{k+1}\right) &= \left(r^{k+1}, r^{k+1}\right) + \beta_k \left(p^k, r^{k+1}\right) \\ &= |r^{k+1}|^2 \qquad \text{for } k \geq 0.\end{aligned}$$

Then by our choice of p^0 we have

$$\left(p^k, r^k\right) = |r^k|^2 \quad \text{for } k \geq 0.$$

This pattern of alternatively using (14.2.1b) and (14.2.1c) will be used repeatedly in our analysis of the conjugate gradient method.

With this last relation and (14.2.2) we have that

$$\alpha_k = \frac{|r^k|^2}{\left(p^k, Ap^k\right)}, \tag{14.2.3}$$

which is a convenient formula for computing α_k. We also have that

$$E\left(x^{k+1}\right) = E\left(x^k\right) - \frac{1}{2}\frac{|r^k|^4}{\left(p^k, Ap^k\right)}.$$

This formula shows that p^k should be chosen so that $\left(p^k, Ap^k\right)$ is minimal, since that will minimize $E\left(x^{k+1}\right)$ given x^k. By (14.2.1c) we see that β_{k-1} should be chosen to minimize $\left(p^k, Ap^k\right)$ given p^{k-1}. We have

$$\left(p^k, Ap^k\right) = \left(r^k, Ar^k\right) + 2\beta_{k-1}\left(r^k, Ap^{k-1}\right) + \beta_{k-1}^2\left(p^{k-1}, Ap^{k-1}\right),$$

and so the optimal choice of β_{k-1} is

$$\beta_{k-1} = -\frac{\left(r^k, Ap^{k-1}\right)}{\left(p^{k-1}, Ap^{k-1}\right)} \quad \text{for } k \geq 1$$

or, equivalently,

$$\beta_k = -\frac{\left(r^{k+1}, Ap^k\right)}{\left(p^k, Ap^k\right)} \quad \text{for } k \geq 0. \tag{14.2.4}$$

Our first conclusion from this formula results from using this formula for β_k with (14.2.1c). We have

$$\left(p^{k+1}, Ap^k\right) = \left(r^{k+1}, Ap^k\right) + \beta_k\left(p^k, Ap^k\right) = 0$$

and so we obtain the important result that

$$\left(p^{k+1}, Ap^k\right) = 0 \quad \text{for } k \geq 0, \tag{14.2.5}$$

which we describe by saying that consecutive search directions are conjugate. Using (14.2.5) with (14.2.1c), we find

$$\begin{aligned}\left(p^k, Ap^k\right) &= \left(r^k, Ap^k\right) + \beta_{k-1}\left(p^{k-1}, Ap^k\right)\\ &= \left(r^k, Ap^k\right),\end{aligned}$$

which we use with (14.2.1b) and (14.2.3) to obtain

$$\begin{aligned}\left(r^{k+1}, r^k\right) &= \left(r^k, r^k\right) - \alpha_k\left(Ap^k, r^k\right)\\ &= \left(r^k, r^k\right) - \alpha_k\left(p^k, Ap^k\right)\\ &= 0.\end{aligned} \tag{14.2.6}$$

We now obtain a more convenient formula for β_k than (14.2.4). First, by (14.2.1b) and (14.2.6),

$$\begin{aligned}\left(r^{k+1}, r^{k+1}\right) &= \left(r^{k+1}, r^k\right) - \alpha_k\left(r^{k+1}, Ap^k\right)\\ &= -\alpha_k\left(r^{k+1}, Ap^k\right),\end{aligned}$$

so by (14.2.4) our formula for β_k is

$$\beta_k = \frac{1}{\alpha_k}\frac{|r^{k+1}|^2}{(p^k, Ap^k)} = \frac{|r^{k+1}|^2}{|r^k|^2}.$$

We now collect the formulas for the conjugate gradient method:

$$p^0 = r^0 = b - Ax^0, \tag{14.2.7a}$$

$$x^{k+1} = x^k + \alpha_k p^k, \tag{14.2.7b}$$

$$r^{k+1} = r^k - \alpha_k Ap^k, \tag{14.2.7c}$$

$$p^{k+1} = r^{k+1} + \beta_k p^k, \tag{14.2.7d}$$

$$\alpha_k = \frac{|r^k|^2}{\left(p^k, Ap^k\right)}, \tag{14.2.7e}$$

$$\beta_k = \frac{|r^{k+1}|^2}{|r^k|^2}. \tag{14.2.7f}$$

The implementation of these formulas in a computer program is discussed in the next section. We conclude this section with some basic observations about the algorithm.

We see from formulas (14.2.7) that if β_k is small, i.e., if $|r^{k+1}|$ is much less that $|r^k|$, then p^{k+1} is essentially r^{k+1} and the conjugate gradient method is close to the steepest descent method. If $|r^{k+1}|$ is not much less than $|r^k|$, then the new search direction, p^{k+1},

will not be close to the local steepest descent direction, r^{k+1}. Notice that the vectors r^k as defined by (14.2.7c) are equal to the residual $b - Ax^k$ for all values of k; see Exercise 14.2.1.

Next we prove a very interesting and significant result about the residuals and search directions for the conjugate gradient method.

Theorem 14.2.1. *For the conjugate gradient method (14.2.7), the residuals and search directions satisfy the relations*

$$\left(r^k, r^j\right) = \left(p^k, Ap^j\right) = 0 \quad \textit{for } k \neq j. \tag{14.2.8}$$

Proof. We prove this result by induction. First notice that

$$\left(r^0, r^1\right) = 0$$

and

$$\left(p^0, Ap^1\right) = 0$$

by (14.2.6) and (14.2.5).

Next, assume that

$$\left(r^\ell, r^j\right) = \left(p^\ell, Ap^j\right) = 0 \quad \text{for } 0 \leq j < \ell \leq k.$$

We wish to show that this holds for all j and ℓ with $0 \leq j < \ell \leq k+1$ as well. First, by (14.2.6) and (14.2.5), we take the case with j equal to k and ℓ equal to $k+1$:

$$\left(r^{k+1}, r^k\right) = \left(p^{k+1}, Ap^k\right) = 0.$$

Now assume that j is less than k. By (14.2.7c) and (14.2.7d) we have

$$\begin{aligned}\left(r^{k+1}, r^j\right) &= \left(r^k, r^j\right) - \alpha_k \left(Ap^k, r^j\right) \\ &= -\alpha_k \left(Ap^k, p^j - \beta_{j-1} p^{j-1}\right) \\ &= 0\end{aligned}$$

since $\left(p^\ell, Ap^j\right)$ and $\left(p^\ell, Ap^{j-1}\right)$ are zero by our induction hypothesis. Also, for j less than k

$$\begin{aligned}\left(p^{k+1}, Ap^j\right) &= \left(r^{k+1}, Ap^j\right) + \beta_k \left(p^k, Ap^j\right) \\ &= \left(r^{k+1}, r^j - r^{j+1}\right) \alpha_j^{-1} \\ &= 0\end{aligned}$$

by the result just proved. This completes the proof of the theorem. □

Theorem 14.2.1 has the following immediate corollary.

Corollary 14.2.2. *If the matrix A is a $K \times K$ symmetric positive definite matrix, then the conjugate gradient algorithm converges in at most K steps.*

Proof. By Theorem 14.2.1 all the residuals are mutually orthogonal, by (14.2.8). Thus r^K is orthogonal to r^k for $k = 0, \ldots, K-1$. Since the dimension of the space is K, r^K must be zero, and so the method must be converged within K steps. □

This corollary is not often of practical importance, since for an elliptic difference equation on the square, e.g., the five-point Laplacian (12.5.1) with grid spacing $\Delta x = \Delta y = 1/N$ the vectors have dimension $K = (N-1)^2$, which is quite large. However, it does turn out that often the conjugate gradient method is essentially converged in far fewer than K steps. When viewed as an iterative method, it is very effective, the number of iteration steps being on the order of N (i.e., $K^{1/2}$) for elliptic difference equations. This is proved in section 14.4.

We have derived the conjugate gradient method by minimizing the quadratic functional $E(y)$ or $F(y)$. Notice that by (14.1.2)

$$\begin{aligned} F(y) &= \tfrac{1}{2}\,(y-x, A(y-x)) \\ &= \tfrac{1}{2}\left(-A^{-1}r, -r\right) = \tfrac{1}{2}\left(r, A^{-1}r\right). \end{aligned}$$

Thus, the conjugate gradient method minimizes the functional $(r, A^{-1}r)$ at each step in the search direction.

Example 14.2.1. Table 14.2.1 displays the results of computations using both the SOR and conjugate gradient methods to solve for the solution of the five-point Laplacian on the unit square with Dirichlet boundary conditions. The exact solution of the partial differential equation was $u = e^x \sin y$ for $0 \le x, y \le 1$. The finite difference grid used equal grid spacing in each direction. The three different grid spacings are displayed in the first column of the table.

For both methods the initial iterate was the grid function that was equal to the exact solution on the boundary and was zero in the interior of the square. The SOR method was terminated when the L^2 norm of the changes to the solution, given by

$$\omega\left(\sum_{\ell,m} |\tfrac{1}{4}(v^n_{\ell+1,m} + v^{n+1}_{\ell-1,m} + v^n_{\ell,m+1} + v^{n+1}_{\ell,m-1}) - v^n_{\ell,m}|^2 h^2\right)^{1/2},$$

was less than the tolerance of 10^{-7}. The sum is for all interior grid points. The value for ω was $2(1+\pi h)^{-1}$ for each case.

The conjugate gradient method was also terminated when the norm of the updates to the solution was less than the tolerance of 10^{-7}. The norm of the updates is given by $h\alpha_k |p^k|$. For each method the number of iterations and the norm of the error are given

Table 14.2.1
Comparison of SOR and conjugate gradient methods.

	SOR		Conjugate gradient		
h	Iterations	Error	Iterations	Error	Residual
0.100	31	5.52–5	27	5.51–5	1.91–8
0.050	64	1.38–5	54	1.39–5	3.19–8
0.025	122	3.21–6	107	3.48–6	2.59–8

for the three values of h equal to $1/10$, $1/20$, and $1/40$. In addition, the norms of the residuals are displayed for the conjugate gradient method.

Table 14.2.1 clearly shows for both methods that the number of iterates is proportional to h^{-1}. The table also demonstrates the second-order accuracy of the five-point Laplacian finite difference scheme. Decreasing the tolerance from 10^{-7} to 10^{-8} and 10^{-9} decreased the residuals for the conjugate gradient method but did not decrease the errors. This shows that the error given is primarily due to the truncation error inherent in the finite difference scheme and is not the error due to the iterative method.

The error shown for h equal to 1/40 for the SOR method actually increased as the tolerance was reduced from 10^{-7} to 10^{-8}. The error shown in the table is due to the fortuitous circumstance that the iterate at which the method was stopped was closer to the solution to the differential equation than it was to the true solution to the difference scheme. When the tolerance was reduced to 10^{-8}, the error was essentially that of the conjugate gradient method. □

In doing computations to demonstrate the order of accuracy of schemes and the speed of iterative methods, we must be careful to distinguish between errors due to the use of finite difference schemes, i.e., truncation errors, and errors due to the iterative method. The results shown in Table 14.2.1 were done in double precision to remove the arithmetic errors due to the finite precision of the computer. Double-precision calculations are often not needed in practical computations because the arithmetic errors are usually much smaller than the uncertainty of the data.

Since the conjugate gradient method is more expensive per step than SOR in terms of both storage and computation time, Table 14.2.1 shows that SOR is more efficient than the conjugate gradient method for this problem. A major advantage of the conjugate gradient method is that it can be easily modified to a preconditioned conjugate gradient method, as is shown in Section 14.5. A second advantage of the conjugate gradient method is that it does not require the user to specify any parameters, such as the iteration parameter ω required by SOR methods.

Exercises

14.2.1. Prove by induction that the vectors r^k as defined by (14.2.7) are equal to the residual $b - Ax^k$ for each k.

14.2.2. A skew matrix A is one for which $A^T = -A$. Show that the following algorithm converges when A is skew and nonsingular:

$$\begin{aligned} p_0 &= -Ar^0, \\ x^{k+1} &= x^k + \alpha_k p^k, \\ r^{k+1} &= r^k - \alpha_k A p^k, \\ p^{k+1} &= -Ar^{k+1} + \beta_k p^k, \\ \alpha_k &= |Ar^k|^2/|Ap^k|^2, \\ \beta_k &= |Ar^{k+1}|^2/|Ar^k|^2. \end{aligned}$$

Hint: Show that α_k and β_{k-1} minimize $|r^k|^2$ at each step. Also show that $\left(r^{k+1},\ Ap^k\right) = \left(Ar^{k+1},\ Ar^k\right) = 0$ for all k.

14.2.3. Show that if A is skew but singular, then with the algorithm given in Exercise 14.2.2, the vectors x^k converge to a vector x^* and the vectors r^k converge to r^*, such that

$$r^* = b - Ax^*$$

and r^* is a null vector of A. *Hint:* Show that $|r^k|$ converges and that $|Ar^k| \leq \|A\|\ |r^k - r^{k+1}|$ and $|r^{k+1} - r^k|^2 = |r^k|^2 - |r^{k+1}|^2$.

14.3 Implementing the Conjugate Gradient Method

We now discuss how to implement the conjugate gradient method using the five-point Laplacian on a uniform grid as an illustration. We begin by considering (14.2.7) and see that four vectors of dimension K are required. These are x^k, r^k, p^k, and an additional vector q^k, which is used to store the values of Ap^k.

We start with an initial iterate x^0 and then compute $r^0 = b - Ax^0$, $q^0 = Ar^0$, and α_0 with $p^0 = r^0$. Then (14.2.7) becomes

$$x^{k+1} = x^k + \alpha_k p^k, \tag{14.3.1a}$$

$$r^{k+1} = r^k - \alpha_k q^k, \tag{14.3.1b}$$

$$p^{k+1} = r^{k+1} + \beta_k p^k, \tag{14.3.1c}$$

$$q^{k+1} = Ar^{k+1} + \beta_k q^k, \tag{14.3.1d}$$

$$\alpha_k = \frac{|r^k|^2}{\left(p^k, q^k\right)}, \tag{14.3.1e}$$

$$\beta_k = \frac{|r^{k+1}|^2}{|r^k|^2}. \tag{14.3.1f}$$

One can avoid using the vectors q^k if Ap^k is computed twice, once for (14.3.1b) and once for evaluating α^k.

We now show what these formulas become for the example of solving Poisson's equation on the unit square with equal spacing in both directions. The vectors will now be indexed by their grid point indices, and we denote the components of the vector x by the grid function $v_{\ell,m}$. The equations to solve are

$$-v_{\ell+1,m} - v_{\ell-1,m} - v_{\ell,m+1} - v_{\ell,m-1} + 4v_{\ell,m} = -h^2 f_{\ell,m}, \tag{14.3.2}$$

which forms the system of equations $Ax = b$. Notice that A is positive definite and symmetric and that the vector b contains both the values $h^2 f_{\ell,m}$ and the values of the solution on the boundary.

First, $v^0_{\ell,m}$ is given and then $r^0_{\ell,m}$ is computed in the interior as

$$\begin{aligned} r^0_{\ell,m} &= -h^2 f_{\ell,m} + v^0_{\ell+1,m} + v^0_{\ell-1,m} + v^0_{\ell,m+1} + v^0_{\ell,m-1} - 4v^0_{\ell,m}, \\ p^0_{\ell,m} &= r^0_{\ell,m}, \end{aligned}$$

with $|r^0|^2$ also being computed. Then $q^0_{\ell,m}$ is computed as

$$q^0_{\ell,m} = 4r^0_{\ell,m} - r^0_{\ell+1,m} - r^0_{\ell-1,m} - r^0_{\ell,m+1} - r^0_{\ell,m-1} \tag{14.3.3}$$

and the inner product $\left(p^0, q^0\right)$ is also computed to evaluate α_0 as $|r^0|^2/(p^0, q^0)$. Note that for Dirichlet boundary data, r^k, p^k, and q^k should be zero on the boundary.

Now begins the main computation loop. First v and r are updated by

$$\begin{aligned} v^{k+1}_{\ell,m} &= v^k_{\ell,m} + \alpha_k p^k_{\ell,m}, \\ r^{k+1}_{\ell,m} &= r^k_{\ell,m} - \alpha_k q^k_{\ell,m}, \end{aligned}$$

with $|r^{k+1}|^2$ also being computed. Using $|r^{k+1}|^2$, β_k is computed; then p and q are updated by

$$\begin{aligned} p^{k+1}_{\ell,m} &= r^{k+1}_{\ell,m} + \beta_k p^k_{\ell,m}, \\ q^{k+1}_{\ell,m} &= 4r^{k+1}_{\ell,m} - r^{k+1}_{\ell+1,m} - r^{k+1}_{\ell-1,m} - r^{k+1}_{\ell,m+1} - r^{k+1}_{\ell,m-1} + \beta_k q^k_{\ell,m}, \end{aligned} \tag{14.3.4}$$

and the inner product (p^{k+1}, q^{k+1}) is computed by accumulating the products $p^{k+1}_{\ell,m} q^{k+1}_{\ell,m}$. Finally, α_{k+1} is computed as the ratio $|r^{k+1}|^2/\left(p^{k+1}, q^{k+1}\right)$ and k is incremented.

It is important to notice that in the computer code there is no need to use variables indexed by the iteration counter k. The values of α_k and β_k are not required beyond the kth iteration, and thus the implementation should use only variables α and β.

A trick can be used to reduce the code that initializes p^0 and q^0. After v^0 and r^0 have been computed, set β equal to zero and then use the code for formulas (14.3.4) to compute p^0 and q^0. This avoids using separate code for (14.3.3) and (14.3.4).

The conjugate gradient method is terminated when either $\alpha_k|p^k|$ or $|r^k|$ is sufficiently small. For most systems these two quantities are good indicators of how close the current iterate x^k is to the true solution. As with the general linear methods, e.g., SOR, the method should be continued until the error in the iteration is comparable to the truncation error in the numerical method. There is no reason to solve the linear system exactly when there is intrinsic truncation error due to using finite difference methods.

It should also be pointed out that it is not wise to compute the residual r^k as $b - Ax^k$, and the formula (14.3.1b) should be used instead. Although r^k is mathematically equivalent to $b - Ax^k$, when using the finite precision of a computer there is a loss of significant digits when the residual is computed as $b - Ax^k$, since the two vectors b and Ax^k will be nearly the same and much larger than r^k when k is not too small. The formula (14.3.1b) avoids this problem.

In those cases where the matrix A is ill conditioned, there will usually be a significant difference between the computed vector r^k and the true residual for large values of k. Nonetheless, the method, as given by (14.3.1), will converge to machine precision even in the presence of these rounding errors. Of course, one must not set the convergence criteria smaller than what can be obtained with the machine arithmetic.

Exercises

14.3.1. Use the conjugate gradient method to solve Poisson's equation

$$u_{xx} + u_{yy} = -4\cos(x + y)\sin(x - y)$$

on the unit square. The boundary conditions and exact solution are given by the formula $u = \cos(x + y)\sin(x - y)$. Use the standard five-point difference scheme with $h = \Delta x = \Delta y = 0.1,\ 0.05,$ and 0.025. The initial iterate should be zero in the interior of the square. Comment on the accuracy of the scheme and the efficiency of the method. Stop the iterative method when the L^2 norm of the change is less than 10^{-6}.

14.3.2. Use the conjugate gradient method to solve Poisson's equation

$$u_{xx} + u_{yy} = -2\cos x \sin y$$

on the unit square. The boundary conditions and exact solution are given by the formula $u = \cos x \sin y$. Use the standard five-point difference scheme with $h = \Delta x = \Delta y = 0.1,\ 0.05,$ and 0.025. The initial iterate should be zero in the interior of the square. Comment on the accuracy of the scheme and the efficiency of the method. Stop the iterative method when the L^2 norm of the change is less than 10^{-6}. Compare with the results of the SOR method applied to this same equation (see Exercise 13.5.1).

14.4 A Convergence Estimate for the Conjugate Gradient Method

Theorem 14.2.1 shows that the conjugate gradient method will converge in at most K steps if A is a $K \times K$ matrix. However, we will now prove an estimate on the rate of convergence of the method that shows that the method is often essentially converged after far fewer than K steps.

Theorem 14.4.1. *If A is a symmetric positive definite matrix whose eigenvalues lie in the interval $[a, b]$, with $0 < a$, then the error vector e^k for the conjugate gradient method satisfies*

$$\left(e^k, Ae^k\right)^{1/2} \leq 2\left(\frac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}}\right)^k \left(e^0, Ae^0\right)^{1/2}. \tag{14.4.1}$$

Proof. We begin with the observation based on (14.2.7b) and (14.2.7c) that the residual after k steps of the conjugate gradient method can be expressed as a linear combination of the set of vectors $\{A^j r^0\}$ for j from 0 to k. We express this observation as

$$r^k = R_k(A)r^0, \tag{14.4.2}$$

where $R_k(\lambda)$ is a polynomial in λ of exact degree k (see Exercise 14.4.1). The coefficients of the polynomial $R_k(\lambda)$ depend on the initial residual r^0. We will also make use of the observation that

$$R_k(0) = 1 \tag{14.4.3}$$

for all nonnegative integers k. (If $A = 0$, then by (14.2.7c), $r^k = r^0$.)

The error e^k on the kth step of the conjugate gradient method is related to the residual by

$$r^k = Ae^k. \tag{14.4.4}$$

Since matrix A commutes with $R_k(A)$, a polynomial in A, we have by (14.4.2) that

$$A\left(e^k - R_k(A)e^0\right) = 0,$$

and since A is nonsingular we have

$$e^k = R_k(A)e^0. \tag{14.4.5}$$

We now use Theorem 14.2.1 to establish that

$$\left(e^k, Ae^k\right) = \left(Q_k(A)e^0, Ae^k\right) \tag{14.4.6}$$

for any polynomial $Q_k(\lambda)$ of degree k satisfying $Q_k(0) = 1$. Relation (14.4.6) is proved as follows. Using (14.4.5) and Theorem 14.2.1, we have

$$\begin{aligned}\left(e^k, Ae^k\right) &= \left(e^k, r^k\right) \\ &= \left(e^k + \sum_{j=0}^{k-1} \gamma_j r^j,\ r^k\right)\end{aligned}$$

for any choice of the coefficients γ_j. But we then have, by (14.4.4) and (14.4.5), that

$$\begin{aligned}e^k + \sum_{j=0}^{k-1} \gamma_j r^j &= \left[R_k(A) + \sum_{j=0}^{k-1} \gamma_j A R_j(A)\right] e^0 \\ &= Q_k(A)e^0,\end{aligned}$$

where it is easy to see that, by appropriate choice of the γ_j, $Q_k(\lambda)$ can be any polynomial of degree k satisfying $Q_k(0) = 1$. This establishes (14.4.6).

We now use the Cauchy–Schwarz inequality for positive definite matrices (see Exercise 14.1.7) to obtain

$$\begin{aligned}\left(e^k, Ae^k\right) &= \left(Q_k(A)e^0, Ae^k\right) \\ &\le \left(Q_k(A)e^0, AQ_k(A)e^0\right)^{1/2} \left(e^k, Ae^k\right)^{1/2},\end{aligned}$$

from which we obtain

$$\left(e^k, Ae^k\right) \le \left(Q_k(A)e^0, AQ_k(A)e^0\right). \tag{14.4.7}$$

We now wish to choose $Q_k(A)$ so that the right-hand side of (14.4.7) is as small as possible, or nearly so. We will actually only estimate the minimum value of the right-hand side. We begin by using the spectral mapping theorem (see Appendix A). Since the eigenvalues of A are in the interval $[a, b]$, we have that

$$\left(Q_k(A)e^0, AQ_k(A)e^0\right) \le \max_{a\le\lambda\le b} |Q_k(\lambda)|^2 \left(e^0, Ae^0\right). \tag{14.4.8}$$

We will choose the polynomial $Q_k(\lambda)$ so that $|Q_k(\lambda)|$ is quite small for λ in $[a, b]$. Recall that $Q_k(0)$ is 1. Based on an understanding of the properties of orthogonal polynomials, we choose

$$Q_k(\lambda) = \frac{T_k\left(\frac{b+a-2\lambda}{b-a}\right)}{T_k\left(\frac{b+a}{b-a}\right)},$$

where $T_k(\mu)$ is the Tchebyshev polynomial of degree k given by

$$T_k(\mu) = \begin{cases} \cos(k \cos^{-1} \mu) & \text{if } |\mu| \le 1 \\ [\text{sign}(\mu)]^k \cosh(k \cosh^{-1} |\mu|) & \text{if } |\mu| \ge 1. \end{cases} \tag{14.4.9}$$

See Exercise 14.4.2. Notice that $Q_k(0)$ is 1.

For λ in the interval $[a, b]$, the value of $|b + a - 2\lambda|/(b - a)$ is bounded by 1 and $|T_k(\mu)|$ for $\mu \in [-1, 1]$ is at most 1; therefore, we have

$$\max_{a \le \lambda \le b} |Q_k(\lambda)| \le \left[T_k \left(\frac{b+a}{b-a} \right) \right]^{-1} = \left[\cosh \left(k \cosh^{-1} \left(\frac{b+a}{b-a} \right) \right) \right]^{-1}.$$

As k increases, the value of $\cosh\{k \cosh^{-1} [(b + a)/(b - a)]\}$ also increases, showing that $\left(e^k, Ae^k\right)$ decreases with k. To obtain a more useful estimate of this quantity, we set

$$\frac{b+a}{b-a} = \cosh \sigma = \frac{e^\sigma + e^{-\sigma}}{2}.$$

Solving this equation for e^σ, we have

$$e^\sigma = \frac{\sqrt{b} + \sqrt{a}}{\sqrt{b} - \sqrt{a}}.$$

(There should be no cause for confusion between e^σ, which is the exponential of σ, and e^k, which is the kth error vector.)

We then obtain

$$\begin{aligned} \cosh k\sigma = \frac{e^{k\sigma} + e^{-k\sigma}}{2} &= \frac{1}{2} \left(\frac{\sqrt{b} + \sqrt{a}}{\sqrt{b} - \sqrt{a}} \right)^k \left[1 + \left(\frac{\sqrt{b} - \sqrt{a}}{\sqrt{b} + \sqrt{a}} \right)^{2k} \right] \\ &\ge \frac{1}{2} \left(\frac{\sqrt{b} + \sqrt{a}}{\sqrt{b} - \sqrt{a}} \right)^k. \end{aligned}$$

Thus we have

$$\max_{a \le \lambda \le b} |Q_k(\lambda)| \le 2 \left(\frac{\sqrt{b} - \sqrt{a}}{\sqrt{b} + \sqrt{a}} \right)^k.$$

This estimate with (14.4.8) gives (14.4.1), which proves Theorem 14.4.1. □

Theorem 14.4.1 shows that the conjugate gradient method converges faster when the eigenvalues of A are clustered together in the sense that a/b is close to 1. Notice also that the estimate (14.4.1) is independent of simple scaling of the matrix A; i.e., the estimate is the same for $Ax = b$ and $\alpha Ax = \alpha b$ for any positive number α. For the five-point Laplacian on the unit square, the value of $\left(\sqrt{b} - \sqrt{a}\right) / \left(\sqrt{b} + \sqrt{a}\right)$ is $1 - O(h)$, as with SOR, and indeed the two methods are about equal in terms of the number of iterations required for a solution, as shown in Table 14.2.1 (see Exercise 14.4.3).

Exercises

14.4.1. Using induction on k, verify relation (14.4.2). You may wish to also show that p^k can be expressed as a polynomial in A multiplying r^0.

14.4.2. Verify that the Tchebyshev polynomials $T_k(\mu)$ given by (14.4.9) are indeed polynomials of degree k. *Hint:* Use the formula

$$\cos(k+1)\theta = -\cos(k-1)\theta + 2\cos k\theta \cos\theta$$

and a similar formula for $\cosh(k+1)\theta$ to establish a recurrence relation between the $T_k(\mu)$.

14.4.3. For the five-point Laplacian on the unit square with equal spacing in each direction, show that $\sqrt{a/b}$ is approximately $\pi h/2$.

14.5 The Preconditioned Conjugate Gradient Method

A technique resulting in further acceleration of the conjugate gradient method is the preconditioned conjugate gradient method. We first discuss this method in some generality and then examine the particular case of preconditioning with SSOR.

The basic idea of the preconditioned conjugate gradient method is to replace the system

$$Ax = b$$

by

$$B^{-1}AB^{-T}\left(B^T x\right) = B^{-1}b,$$

where $B^{-1}AB^{-T}$ is a matrix for which the conjugate gradient method converges faster than it does with A itself. Matrix B is chosen so that computing $B^{-T}y$ and $B^{-1}y$ are easy operations to perform. Note that $B^{-1}AB^{-T}$ is symmetric and positive definite when A is.

According to Theorem 14.4.1, to get faster convergence, we wish to have the eigenvalues of $B^{-1}AB^{-T}$ more clustered together than are those of A. Since A is symmetric and positive definite, there is a matrix C so that $A = CC^T$, and B is usually chosen to approximate C in some sense. Note also that B need only approximate a multiple of C, so that $B^{-1}AB^{-T}$ is closer to being a multiple of the identity than is A itself.

Consider now the conjugate gradient method applied to

$$\tilde{A}\tilde{x} = \tilde{b},$$

where

$$\tilde{A} = B^{-1}AB^{-T}, \quad \tilde{x} = B^T x, \quad \tilde{b} = B^{-1}b.$$

We have from (14.2.7) that

$$\begin{aligned} \tilde{x}^{k+1} &= \tilde{x}^k + \alpha_k \tilde{p}^k, \\ \tilde{r}^{k+1} &= \tilde{r}^k - \alpha_k \tilde{A}\tilde{p}^k, \\ \tilde{p}^{k+1} &= \tilde{r}^{k+1} + \beta_k \tilde{p}^k, \end{aligned} \tag{14.5.1}$$

where $\alpha_k = |\tilde{r}^k|^2/(\tilde{p}^k, \tilde{A}\tilde{p}^k)$ and $\beta_k = |\tilde{r}^{k+1}|^2/|\tilde{r}^k|^2$.

Now let us rewrite (14.5.1) in terms of the original variables x rather than $\tilde{x}$. Using $x^k = B^{-T}\tilde{x}^k$, $p^k = B^{-T}\tilde{p}^k$, and $r^k = B\tilde{r}^k$, we have

$$\begin{aligned} x^{k+1} &= x^k + \alpha_k p^k, \\ r^{k+1} &= r^k - \alpha_k A p^k, \\ p^{k+1} &= M^{-1} r^{k+1} + \beta_k p^k, \end{aligned}$$

where $M = BB^T$ and

$$\alpha_k = \frac{\left(r^k, M^{-1}r^k\right)}{\left(p^k, Ap^k\right)}, \quad \beta_k = \frac{\left(r^{k+1}, M^{-1}r^{k+1}\right)}{\left(r^k, M^{-1}r^k\right)}.$$

We see that the effect of the preconditioning is to alter the equation for updating the search direction p^{k+1} and to alter the definitions of α_k and β_k. For the method to be effective, we must easily be able to solve

$$r = Mz = BB^T z$$

for z. A common choice of B is to take $B = \tilde{L}$, where $\tilde{L}$ is an approximate lower triangular factor of A in the sense that

$$A = \tilde{L}\tilde{L}^T + N,$$

where N is small in some sense.

Preconditioning by SSOR

We now consider SSOR and show how it can be used as a preconditioning for the conjugate gradient method. We assume that A can be written in the form

$$A = I - L - L^T.$$

Notice that the matrix A in (14.3.2) is actually in the form $4(I - L - L^T)$, but the scalar multiple does not affect the conclusions. SSOR is a two-step process given by

$$\begin{aligned} v^{k+1/2} &= v^k + \omega\left(Lv^{k+1/2} + L^T v^k - v^k + b\right), \\ v^{k+1} &= v^{k+1/2} + \omega\left(Lv^{k+1/2} + L^T v^{k+1} - v^{k+1/2} + b\right). \end{aligned} \tag{14.5.2}$$

We wish to rewrite this in the form $M(v^{k+1} - v^k) = r^k$. Notice that we can express $v^{k+1} - v^k$ as a linear function of r^k in this way because the construction of v^{k+1} is linear, and if r^k were zero, then the update $v^{k+1} - v^k$ would also be zero. It remains to determine the matrix M and to determine if it has the form $\tilde{B}\tilde{B}^T$.

We rewrite the first step as

$$v^{k+1/2} - v^k - \omega L\left(v^{k+1/2} - v^k\right) = \omega\left(Lv^k + L^T v^k - v^k + b\right) = \omega r^k.$$

We can therefore write

$$v^{k+1/2} = v^k + (I - \omega L)^{-1}\,\omega r^k. \tag{14.5.3}$$

The second step of (14.5.2) can be rewritten as

$$\left(I - \omega L^T\right) v^{k+1} = \left(I(1-\omega) + \omega L\right) v^{k+1/2} + \omega b,$$

and substituting from (14.5.3) we have

$$\left(I - \omega L^T\right) v^{k+1} = \left[(1-\omega)I + \omega L\right] v^k + \left[(1-\omega)I + \omega L\right](I - \omega L)^{-1}\,\omega r^k + \omega b$$

or

$$\begin{aligned}
&\left(I - \omega L^T\right)\left(v^{k+1} - v^k\right) \\
&\qquad = \left(-\omega I + \omega L + \omega L^T\right) v^k + \omega b + \left[(1-\omega)I + \omega L\right](I - \omega L)^{-1}\,\omega r^k \\
&\qquad = \omega r^k + \left[(1-\omega)I + \omega L\right](I - \omega L)^{-1}\,\omega r^k \\
&\qquad = (I - \omega L)^{-1}\left[I - \omega L + (1-\omega)\,I + \omega L\right] r^k \\
&\qquad = (I - \omega L)^{-1}\,(2-\omega)\,\omega r^k.
\end{aligned}$$

We thus have

$$\frac{1}{\omega\,(2-\omega)}\,(I - \omega L)\left(I - \omega L^T\right)\left(v^{k+1} - v^k\right) = r^k. \tag{14.5.4}$$

If we compare expression (14.5.4) with the identity

$$A\left(v - v^k\right) = r^k,$$

we see that SSOR can be viewed as an iterative method, which approximates A by the matrix in (14.5.4). Since the matrix in (14.5.4) is in the form BB^T, it is natural to employ the preconditioned conjugate gradient method with $B = (\omega(2-\omega))^{-1/2}\,(I - \omega L)$.

It is important to note that if we are going to use SSOR alone to solve the problem, we would use (14.5.2) with immediate replacement. Formula (14.5.4) is important only when using SSOR as a preconditioner.

We now apply this preconditioning matrix to Laplace's equation in a square. We have

$$\begin{aligned} x^{k+1} &= x^k + \alpha_k p^k, \\ r^{k+1} &= r^k - \alpha_k A p^k, \end{aligned} \tag{14.5.5}$$

and

$$p^{k+1} = z^{k+1} + \beta_k p^k,$$

where z^{k+1} is computed using (14.5.4). The computation of z^{k+1} is implemented as follows:

$$\begin{aligned} \tilde{z}_{\ell,m}^{k+1} &= \tfrac{1}{4}\omega\left(\tilde{z}_{\ell,m-1}^{k+1} + \tilde{z}_{\ell-1,m}^{k+1}\right) + \omega\,(2-\omega)\, r_{\ell,m}^{k+1}, \\ z_{\ell,m}^{k+1} &= \tfrac{1}{4}\omega\left(z_{\ell,m+1}^{k+1} + z_{\ell+1,m}^{k+1}\right) + \tilde{z}_{\ell,m}^{k+1} \end{aligned} \tag{14.5.6}$$

for all interior points with $\tilde{z}$ and z being zero on the boundaries. Notice that the first of these relations should be executed in the order of increasing indices, and the second should be done in the order of decreasing indices.

Notice that the quantities z and $\tilde{z}$ can occupy the same storage locations. The parameters for the preconditioned method are computed by the formulas

$$\alpha_k = \frac{\left(r^k, z^k\right)}{\left(p^k, Ap^k\right)},$$

$$\beta_k = \frac{\left(r^{k+1}, z^{k+1}\right)}{\left(r^k, z^k\right)}.$$

The method of (14.5.6) is a method for solving

$$(I - \omega L)\,\tilde{z}^{k+1} = \omega\,(2-\omega)\, r^{k+1},$$

$$\left(I - \omega L^T\right) z^{k+1} = \tilde{z}^{k+1}.$$

Other ways of computing z^{k+1} can also be used. Notice that we have scaled $\tilde{z}$ to avoid taking the square root of $\omega(2-\omega)$. We can also dispense with the factor $\omega(2-\omega)$, since it represents only a scaling factor.

To implement the preconditioning requires two more loops than does the regular conjugate gradient method. The additional loops, given by (14.5.6), are very simple, and the slight extra effort is more than justified by the substantial increase in speed of the preconditioned method.

We now collect the formulas for implementing the preconditioned conjugate gradient method. To initialize the preconditioned conjugate gradient method, we use

$$p^0 = z^0 = M^{-1} r^0,$$

as we see from the relations between p^0, $\tilde{p}^0$, r^0, and $\tilde{r}^0$. The formulas are:

$$x^{k+1} = x^k + \alpha_k p^k, \tag{14.5.7a}$$

$$r^{k+1} = r^k - \alpha_k q^k, \tag{14.5.7b}$$

$$z^{k+1} = M^{-1} r^{k+1}, \tag{14.5.7c}$$

$$p^{k+1} = z^{k+1} + \beta_k p^k, \tag{14.5.7d}$$

$$q^{k+1} = A z^{k+1} + \beta_k q^k, \tag{14.5.7e}$$

$$\alpha_k = \frac{\left(r^k, z^k\right)}{\left(p^k, q^k\right)}, \tag{14.5.7f}$$

$$\beta_k = \frac{\left(r^{k+1}, z^{k+1}\right)}{\left(r^k, z^k\right)}. \tag{14.5.7g}$$

As with (14.3.1), we can avoid using the vectors q^k if Ap^k is computed twice, once for (14.5.7b) and once for evaluating α^k in (14.5.7f).

The preconditioned conjugate gradient method can be significantly faster than the conjugate gradient method. As we can see, it requires only minor modifications to a conjugate gradient method to implement a preconditioned conjugate gradient method. The choice of ω in the SSOR preconditioner is not as critical as it is in the SSOR method itself. The spectral radius for the preconditioned conjugate gradient method with the SSOR preconditioner is $1 - O\left(N^{-1/2}\right)$. This is illustrated in Table 14.5.1.

Example 14.5.1. Table 14.5.1 shows the results of solving Poisson's equation using the point SOR method, the conjugate gradient method, and the preconditioned conjugate gradient method, with SSOR as the preconditioner. The exact solution that was calculated was $u = \cos x \sin y$ for $0 \le x, y \le 1$. The finite difference grid used equal grid spacing in each direction. The three different grid spacings are displayed in the first column of the table. The next columns show the numbers of iterations required to obtain a converged solution.

For each method the initial iterate was the grid function that was equal to the exact solution on the boundary and was zero in the interior of the square. Each method was terminated when the L^2 norm of the change in the solution was less than 10^{-7}. This convergence criterion was sufficient to produce results for which the error was primarily due to the truncation error. For both the SOR method and the SSOR preconditioner, the value of ω was $2(1 + \pi h)^{-1}$. The table shows that the number of iterations for the first two methods is roughly proportional to h^{-1}, whereas for the preconditioned conjugate gradient method, the number of iterations is proportional to $h^{-1/2}$. These results are similar to those of Table 14.2.1.

Since the work in one iteration of the SOR method is less than that in one iteration of the other two methods, it is not appropriate to judge the methods solely on the number of iterations. The conjugate gradient method involves roughly twice as much work per

Table 14.5.1
Comparison of the speeds of SOR, the conjugate gradient method, and the preconditioned conjugate gradient method.

h	SOR	C.G.	P.C.G.
0.100	33	26	12
0.050	60	52	16
0.025	115	103	22

iteration as does point SOR, and the preconditioned conjugate gradient method involves three to four times as much work as SOR. Thus the preconditioned conjugate gradient method is faster than SOR for h equal to $1/40$, but probably not for the grid spacing of $1/10$. Of course, for even smaller values of the grid spacing h, the preconditioned conjugate gradient method would be even faster relative to SOR. In terms of computer storage, the SOR method requires much less storage than the other two methods, but this is not a significant concern in many scientific computations. □

Formulas (14.5.7) show that five vectors are required to implement the preconditioned conjugate gradient method, as opposed to only four vectors for the conjugate gradient method. One way of using only four vectors for the preconditioned conjugate gradient method is to work with $\tilde{r} = B^{-1}r$ rather than r; see Eisenstat [16]. We then obtain the algorithm

$$x^{k+1} = x^k + \alpha_k p^k, \tag{14.5.8a}$$

$$\tilde{r}^{k+1} = \tilde{r}^k - \alpha_k B^{-1} q^k, \tag{14.5.8b}$$

$$p^{k+1} = B^{-T}\tilde{r}^{k+1} + \beta_k p^k, \tag{14.5.8c}$$

$$q^{k+1} = Ap^{k+1}, \tag{14.5.8d}$$

$$\alpha_k = |\tilde{r}^k|^2 / \left(p^k, q^k\right), \tag{14.5.8e}$$

$$\beta_k = |\tilde{r}^{k+1}|^2 / |\tilde{r}^k|^2. \tag{14.5.8f}$$

The results of the calculations of the vectors $B^{-1}q^k$ and $B^{-T}\tilde{r}^{k+1}$ are stored in the vector q.

Preconditioning by Approximate Cholesky Factorization

Other preconditioning matrices for the conjugate gradient method can be obtained by approximating A as $\tilde{L}\tilde{L}^T$ for a convenient form of $\tilde{L}$. A factorization of a matrix as LL^T, where L is a lower triangular matrix, is called a Cholesky factorization; thus the product $\tilde{L}\tilde{L}^T$ is called an approximate Cholesky factorization of A. As an example of an approximate Cholesky factorization for the matrix for the five-point Laplacian, consider a matrix

$\tilde{L}$ of the form

$$\left(\tilde{L}v\right)_{\ell,m} = av_{\ell,m} + bv_{\ell-1,m} + cv_{\ell,m-1}, \tag{14.5.9}$$

where a, b, and c are constants. It is easy to see then that

$$\left(\tilde{L}^T v\right)_{\ell,m} = av_{\ell,m} + bv_{\ell+1,m} + cv_{\ell,m+1} \tag{14.5.10}$$

if we use the natural ordering of the components $v_{\ell,m}$ in the vector v. We then have, by (14.5.9) and (14.5.10),

$$\begin{aligned}
\left(\tilde{L}\tilde{L}^T v\right)_{\ell,m} &= a\left(\tilde{L}^T v\right)_{\ell,m} + b\left(\tilde{L}^T v\right)_{\ell-1,m} + c\left(\tilde{L}^T v\right)_{\ell,m-1} \\
&= a\left(av_{\ell,m} + bv_{\ell+1,m} + cv_{\ell,m+1}\right) \\
&\quad + b\left(av_{\ell-1,m} + bv_{\ell,m} + cv_{\ell-1,m+1}\right) \\
&\quad + c\left(av_{\ell,m-1} + bv_{\ell+1,m-1} + cv_{\ell,m}\right) \\
&= \left(a^2 + b^2 + c^2\right) v_{\ell,m} + abv_{\ell+1,m} + acv_{\ell,m+1} \\
&\quad + abv_{\ell-1,m} + acv_{\ell,m-1} + bcv_{\ell-1,m+1} \\
&\quad + bcv_{\ell+1,m-1}.
\end{aligned}$$

To have $\tilde{L}\tilde{L}^T$ approximate A, where A corresponds to the five-point Laplacian, we may set

$$a^2 + b^2 + c^2 = 1,$$

$$ab = ac = -\tfrac{1}{4}.$$

The two terms in $\tilde{L}\tilde{L}^T v$ for the terms with subscripts $(\ell - 1, m + 1)$ and $(\ell + 1, m - 1)$ do not match with terms in the five-point Laplacian. They represent the error in the approximation of A by $\tilde{L}\tilde{L}^T$.

Solving the equations for a, b, and c we have

$$a = \frac{\sqrt{2+\sqrt{2}}}{2},$$

$$b = c = -\frac{1}{4a}.$$

To implement this method it is often convenient to approximate A by $\tilde{L}D\tilde{L}^T$, where D is a diagonal matrix. For our particular choice of $\tilde{L}$, D is just a^2 times the identity. Using this choice of $\tilde{L}$ the preconditioned conjugate gradient method is (14.5.5), with (14.5.6) replaced by

$$\begin{aligned}
\hat{z}^{k+1}_{\ell,m} &= \left(\hat{z}^{k+1}_{\ell,m-1} + \hat{z}^{k+1}_{\ell-1,m} + 4r^{k+1}_{\ell,m}\right) d, \\
z^{k+1}_{\ell,m} &= \left(z^{k+1}_{\ell,m+1} + z^{k+1}_{\ell+1,m}\right) d + \hat{z}^{k+1}_{\ell,m},
\end{aligned} \tag{14.5.11}$$

Table 14.5.2
Comparison of three preconditioning methods for the nine-point Laplacian.

h	None	Cholesky	Five-point	Nine-point
0.100	28	16	18	16
0.050	57	28	25	23
0.025	112	52	34	32

where

$$d = \left(2 + \sqrt{2}\right)^{-1}.$$

The temporary variable $\hat{z}$ is defined by

$$\hat{z} = \frac{2}{\sqrt{2+\sqrt{2}}} \tilde{L}^T z.$$

We can try more sophisticated choices for the matrix $\tilde{L}$. For the discrete Laplacian on a square, the preceding methods all do quite well. For matrices arising from other problems we may have to work quite hard to get a good preconditioning matrix.

Example 14.5.2. To solve the difference equations for the fourth-order accurate Poisson's equation (12.5.4), we can use preconditioning based on the five-point Laplacian. This is a simple way to accelerate the solution procedure and it does not affect the accuracy of the scheme. In fact, using SSOR based on the nine-point Laplacian as the preconditioner with the nine-point Laplacian does not give a significant improvement over that using SSOR based on the five-point Laplacian as the preconditioner. This is illustrated in Table 14.5.2.

Table 14.5.2 displays the results of solving Laplace's equation using the nine-point Laplacian with the conjugate gradient method and with three different preconditioning methods. The three preconditioning methods are the approximate Cholesky factorization (14.5.11) for the five-point Laplacian, the SSOR preconditioning using the five-point Laplacian, and the SSOR preconditioning using the nine-point Laplacian. The exact solution that was calculated was $u = e^{3x} \sin 3y$ for $0 \le x, y \le 1$. The finite difference grid used equal grid spacing in each direction. The three different grid spacings are displayed in the first column of the table. The next columns show the number of iterations required to obtain a converged solution. For each method the initial iterate was the grid function that was equal to the exact solution on the boundary and was zero in the interior of the square.

Each method was terminated when the L^2 norm of the change in the solution was less than 10^{-10}. This convergence criterion was sufficient to produce results for which the error was primarily due to the truncation error, similar to the results shown in Table 12.5.1. For both of the SSOR preconditioners, the value of ω was $2(1 + \pi h)^{-1}$. The table shows that the number of iterations for the last two methods is roughly proportional to $h^{-1/2}$. There is not a significant difference between the last two methods, but the nine-point preconditioner is better, as would be expected. The approximate Cholesky method based on the five-point scheme is not as good as the other two methods, but it still offers a significant improvement over the basic conjugate gradient method. □

As Example 14.5.2 illustrates, a preconditioner based on the five-point Laplacian may be a good preconditioner for the nine-point Laplacian. The reason for this is that they are both related to the same partial differential equation. In general, there is a trade-off between the effort it takes to find a better preconditioner and the perhaps marginal increase in performance.

Currently, there is an extensive literature on preconditioning methods. For equations other than those discussed here, one should check the literature to see what methods other researchers have employed.

Exercises

14.5.1. Repeat the calculations of Exercise 14.3.1, but using the preconditioned conjugate gradient method with the SSOR preconditioning. Comment on the efficiency of the method and observe that the number of iterations increases as $O(N^{1/2})$.

14.5.2. Repeat the calculations of Exercise 14.3.2, but using the preconditioned conjugate gradient method with the SSOR preconditioning. Comment on the efficiency of the method and observe that the number of iterations increases as $O(N^{1/2})$.

14.5.3. Repeat the calculations of Exercise 14.3.1, but using the preconditioned conjugate gradient method with the approximate Cholesky factorization as the preconditioning. Comment on the efficiency of the method and observe that the number of iterations increases as $O(N^{1/2})$.

14.5.4. Repeat the calculations of Exercise 14.3.2, but using the preconditioned conjugate gradient method with the approximate Cholesky factorization as the preconditioning. Comment on the efficiency of the method and observe that the number of iterations increases as $O(N^{1/2})$.