7. Conjugate gradients

7.1 A–orthogonal vectors

Throughout this chapter matrix $A = A(n \times n)$ is exclusively assumed to be positive definite and symmetric. We know by now that it is of considerable interest to have matrix $P = P(n \times m)$ so that $P^TAP = D$ is diagonal, one reason being that if matrix P is square and D nonsingular, then $A^{-1} = PD^{-1}P^T$. Columns $p_1, p_2, \ldots, p_n$ of matrix P in the transformation $P^TAP = D$ are A-orthogonal; $p_i^TAp_j = 0$ if $i \neq j$. Such are the eigenvectors of A, which are not only A-orthogonal but moreover I-orthogonal. But eigenvectors are expensive to compute. The triangular factorization $A = L^TDL$ of A also produces A-orthogonal vectors inasmuch as $L^{-T}AL^{-1} = D$.

In this chapter we explore the remarkable properties of a simple algorithm of recent origin, dating from 1950, for the recursive generation of A-orthogonal vectors one after another with only three vectors involved in each recursive step.

Definition. *Let A be a positive definite and symmetric matrix. Vectors p and q are A–orthogonal or A–conjugate if $p^TAq = q^TAp = 0$. I–orthogonal vectors are just orthogonal.*

Theorem 7.1. *The repetitive algorithm*

$$p_0 = r_0 \neq o \ , \ \textit{arbitrary}$$

$$\begin{aligned} \alpha_i &= \frac{r_i^T r_i}{p_i^T A p_i} = \frac{p_i^T r_i}{p_i^T A p_i} \\ r_{i+1} &= r_i - \alpha_i A p_i \\ \beta_i &= -\frac{r_{i+1}^T A p_i}{p_i^T A p_i} = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i} \\ p_{i+1} &= r_{i+1} + \beta_i p_i \end{aligned} \tag{7.1}$$

in which matrix A is a symmetric and positive definite generates vector sequences $r_0, r_1, \ldots, r_m$ and $p_0, p_1, \ldots, p_m$ such that

$$\begin{aligned} p_i^T A p_j &= 0 \\ r_i^T r_j &= 0 \end{aligned} \quad i \neq j \ . \tag{7.2}$$

Proof. By induction. We readily verify that $r_0^T r_1 = 0$. Suppose that $r_i^T r_j = 0 \ i \neq j$, and $p_i^T A p_j = 0 \ i \neq j$ hold for $r_0, r_1, \ldots, r_k$ and $p_0, p_1, \ldots, p_{k-1}$. Under these circumstances

$$r_i^T A p_i = p_i^T A r_i = p_i^T A (p_i - \beta_{i-1} p_{i-1}) = p_i^T A p_i, \ i = 0, 1, \ldots, k-1. \tag{7.3}$$

The next vector p_k is given by

$$p_k = r_k + \beta_{k-1} p_{k-1} \ , \ \beta_{k-1} = -\frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}} \tag{7.4}$$

and $p_{k-1}^T A p_k = 0$. Our assumption that $p_j^T A p_{k-1} = 0, \ j < k-1$ leads to

$$p_j^T A p_k = p_j^T A r_k + \beta_{k-1} p_j^T A p_{k-1} = p_j^T A r_k \ , \ j = 0, 1, \ldots, k-2. \tag{7.5}$$

But

$$A p_j = \frac{1}{\alpha_j}(r_j - r_{j+1}) \tag{7.6}$$

and

$$p_j^T A p_k = r_k^T A p_j = \frac{1}{\alpha_j}(r_k^T r_j - r_k^T r_{j+1}) = 0 \quad j = 0, 1, \ldots, k-2 \tag{7.7}$$

since $r_i^T r_j = 0 \ i, j = 0, 1, \ldots, k \ i \neq j$. The newly constructed p_k is A–orthogonal to all previous $p_0, p_1, \ldots, p_{k-1}$. The next vector r_{k+1} is given by

$$r_{k+1} = r_k - \alpha_k A p_k \ , \ \alpha_k = \frac{r_k^T r_k}{r_k^T A p_k} = \frac{r_k^T r_k}{p_k^T A p_k} \tag{7.8}$$

and $r_{k+1}^T\ r_k = 0$. To prove that $r_{k+1}^T r_j = 0\ j = 0, 1, 2, \ldots, k-1$ we form

$$r_j^T r_{k+1} = r_j^T r_k - \alpha_k r_j^T A p_k = -\alpha_k r_j^T A p_k \tag{7.9}$$

and substitute into it $r_j = p_j - \beta_{j-1} p_{j-1}$,

$$r_j^T A p_k = p_k^T A(p_j - \beta_{j-1} p_{j-1}) = 0 \quad j = 1, 2, \ldots, k-1 \tag{7.10}$$

because $p_i^T A p_j = 0\ i \neq j$.

Hence starting with $r_1^T r_0 = 0$ we have that $r_2^T r_1 = r_2^T r_0 = 0$ and so on, or generally $r_i^T r_j = 0\ i \neq j$; and $p_1^T A p_0 = 0,\ p_2^T A p_1 = p_2^T A p_0 = 0$, or generally $p_i^T A p_j = 0\ i \neq j$.

We have but to prove that $p_i^T r_i = r_i^T r_i$ and that $-r_{i+1}^T A p_i / p_i^T A p_i = r_{i+1}^T r_{i+1} / r_i^T r_i$.

From $p_j = r_j + \beta_{j-1} p_{j-1}$ it results that

$$p_j = r_j + \gamma_{j-1} r_{j-1} + \cdots + \gamma_0 r_0 \tag{7.11}$$

where

$$\gamma_{j-1} = \beta_{j-1},\ \gamma_{j-2} = \beta_{j-1}\beta_{j-2}, \ldots, \gamma_0 = \beta_{j-1} \cdots \beta_1 \beta_0 \tag{7.12}$$

and since $r_j^T r_i = 0$ if $i < j$, then $r_j^T p_j = r_j^T r_j$.

From $r_{i+1} = r_i - \alpha_i A p_i$ we have that

$$r_{i+1}^T r_{i+1} = -\alpha_i r_{i+1}^T A p_i = -\frac{r_i^T r_i}{p_i^T A p_i} r_{i+1}^T A p_i \tag{7.13}$$

and $r_{i+1}^T r_{i+1} / r_i^T r_i = -r_{i+1}^T A p_i / p_i^T A p_i$. End of proof.

Corollary 7.2. *Among vectors p_i and r_i $i = 1, 2, \ldots, m$ generated by the algorithm of Theorem 7.1, the following relationships hold:*

1. $p_j = \|r_j\|^2 \Big(\frac{1}{\|r_0\|^2} r_0 + \frac{1}{\|r_1\|^2} r_1 + \cdots + \frac{1}{\|r_j\|^2} r_j \Big)$.
2. $p_i^T r_j = 0\ ,\ i < j$.
3. $p_i^T r_j = r_i^T r_i\ ,\ i \geq j$.
4. $r_i^T A p_j = 0\ ,\ |i - j| > 1$.
5. $r_i^T A p_i = p_i^T A p_i$.

6. $p_i^T r_0 = p_i^T r_1 = \cdots = p_i^T r_i = r_i^T r_i$.

Proof.

1. Results from a recursive application of

$p_j = r_j + \beta_{j-1} p_{j-1},\ \beta_{j-1} = r_j^T r_j / r_{j-1}^T r_{j-1}$.

2. An immediate result of 1.

3. An immediate result of 1.

4. Results from $r_{i+1} = r_i - \alpha_i A p_i$ and $r_i^T r_j = 0\ i \neq j$.

5. A direct consequence of $r_{i+1} = r_i - \alpha_i A p_i$.

6. An immediate result of 1.

End of proof.

Theorem 7.3. *Vectors p_i and r_i generated in Theorem 7.1 are such that:*

1. $\|p_i\|^2 = \|r_i\|^4 \Big(\dfrac{1}{\|r_0\|^2} + \cdots + \dfrac{1}{\|r_i\|^2} \Big)$.
2. $p_i^T p_j = \|r_j\|^2 \|p_i\|^2 / \|r_i\|^2\ ,\ i \leq j$.
3. $\|r_i\| \leq \|p_i\|$.

Proof.

1. An immediate consequence of statement 1, Corollary 7.2.

2. An immediate consequence of statement 1, Corollary 7.2 and the previous statement.

3. From statement 1 we have that

$$\|p_i\| = \|r_i\| \Big(\frac{\|r_i\|^2}{\|r_0\|^2} + \cdots + 1 \Big)^{1/2} \tag{7.14}$$

and hence $\|p_i\| / \|r_i\| \geq 1$. End of proof.

The following has far-reaching consequences and we devote a separate theorem to it:

Theorem 7.4. *Vectors r_i generated in Theorem 7.1 are such that:*

$$r_i^T A r_j = 0\ ,\ |i - j| > 1. \tag{7.15}$$

Proof. Write

$$r_i = p_i - \beta_{i-1} p_{i-1} \ , \ r_j = p_j - \beta_{j-1} p_{j-1} \tag{7.16}$$

to have

$$r_j^T A r_i = p_i^T A p_j - \beta_{i-1} p_j^T A p_{i-1} - \beta_{j-1} p_{j-1}^T \ p_i + \beta_{i-1}\beta_{j-1} p_{j-1}^T A p_{i-1} \tag{7.17}$$

which equals zero if $i \neq j,\ i-1 \neq j$, and $j-1 \neq i$. End of proof

7.2 Inversion and tridiagonalization

The iterative algorithm in Theorem 7.1 comes to an end when $r_m = o$.

Lemma 7.5. *Let $v_1, v_2, \ldots, v_n$ be the n orthonormal eigenvectors of the symmetric and positive definite A, and let V^m be the subspace of R^n that is spanned by $v_1, v_2, \ldots, v_m$. Then, if $r_0 \in V^m$ so are r_j and p_j.*

Proof. If r_0 and p_0 are in V^m, then so is Ap_0, and consequently r_1 and p_1, and so on. End of proof.

Theorem 7.6. *Let V^m be the subspace of Lemma 7.4. If $r_0 \in V^m$, then $r_m = o$.*

Proof. Since $r_0, r_1, \ldots, r_{m-1}$ are orthogonal in an m-dimensional vector space, the next orthogonal is zero; there are no more than m nonzero orthogonal vectors in V^m. End of proof.

Theorem 7.7. *If the symmetric and positive definite matrix A has m distinct eigenvalues, then $r_m = o$ for any initial $r_0 \neq o$.*

Proof. Expanded in terms of the n orthonormal eigenvectors $v_1, v_2, \ldots, v_n$ of A

$$p_0 = r_0 = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n \tag{7.18}$$

and we group it as

$$p_0 = r_0 = w_1 + w_2 + \cdots + w_m \tag{7.19}$$

where $Aw_i = \lambda_i w_i,\ Aw_j = \lambda_j w_j,\ \lambda_i \neq \lambda_j$. Every r_j and p_j created by the algorithm of Theorem 7.1 is in the m dimensional space spanned by $w_1, w_2, \ldots, w_m$, and since the r's are orthogonal by the argument of the previous theorem $r_m = o$. End of proof.

Theorem 7.8. *If $p_0, p_1, \ldots, p_{n-1}$ are nonzero A–orthogonal vectors, then*

$$A^{-1} = \frac{1}{p_0^T A p_0} p_0 p_0^T + \frac{1}{p_1^T A p_1} p_1 p_1^T + \cdots + \frac{1}{p_{n-1}^T A p_{n-1}} p_{n-1} p_{n-1}^T. \tag{7.20}$$

Proof. Write $P = [p_0\ p_1 \ldots p_{n-1}]$. Then $P^T A P = D$ is diagonal with $D_{ii} = p_i^T A p_i$. Since A is positive definite $D_{ii} > 0$ and $P^{-1} A^{-1} P^{-T} = D^{-1}$ or $A^{-1} = P D^{-1} P^T$, which when written out is the equation in the theorem. End of proof.

Theorem 7.9. *Let $R = [r_0\ r_1 \ldots r_{m-1}]$. Then*

$$R^T A R = T(m \times m),\ R^T R = D(m \times m) \tag{7.21}$$

where T is tridiagonal and D is diagonal.

Proof. This is an immediate consequence of theorems 7.1 and 7.4. End of proof.

The choice of $r_0, r_1, \ldots, r_{m-1}$ as basis for V^m in the Ritz method results in the $m \times m$ eigenproblem

$$Ty = \mu D y \tag{7.22}$$

for tridiagonal $T = R^T A R$ and diagonal $D = R^T R$. Reduction of the Ritz eigenproblem to that of a tridiagonal matrix is due to *Lanczos* to whom we owe the recursive scheme of Theorem 7.1. Deeper discussion of the Lanczos method is deferred to the next chapter.

exercises

7.2.1. Let $Q = [q_1\ q_2\ \ldots\ q_n]$ be such that for given $A = A^T (n \times n)$, $T = Q^T A Q$ is tridiagonal. Write $QT = AQ$,

$$\begin{bmatrix} & & \\ q_1 & q_2 & q_3 \\ & & \end{bmatrix} \begin{bmatrix} \alpha_1 & \beta_2 & \\ \beta_2 & \alpha_2 & \beta_3 \\ & \beta_3 & \alpha_2 \end{bmatrix} = \begin{bmatrix} & & \\ Aq_1 & Aq_2 & Aq_3 \\ & & \end{bmatrix}$$

and show that the choice of q_1 determines Q and T, if $\beta_j > 0$.

7.3 Iterative solution of $Ax = f$

Let $Ax = f$ be a system of linear equations with a positive definite and symmetric matrix A. If we choose an arbitrary vector x_0 and in Theorem 7.1 set $p_0 = r_0 = f - Ax_0$, then we

find that

$$\begin{aligned} r_1 &= r_0 - \alpha_0 A p_0 \\ &= f - A(x_0 + \alpha_0 p_0) \\ &= f - A x_1 \end{aligned} \tag{7.23}$$

where $x_1 = x_0 + \alpha_0 p_0$. Hence the *conjugate gradient* algorithm:

$$\begin{aligned} p_0 = r_0 = f - A x_0 \quad &, \quad x_0 \text{ arbitrary} \\ \alpha_i &= r_i^T r_i / p_i^T A p_i \\ x_{i+1} &= x_i + \alpha_i p_i \\ r_{i+1} &= r_i - \alpha_i A p_i \\ \beta_i &= r_{i+1}^T r_{i+1} / r_i^T r_i \\ p_{i+1} &= r_{i+1} + \beta_i p_i \end{aligned} \tag{7.24}$$

for the iterative solution of $Ax = f$. Since in this algorithm $r_0 = f - Ax_0$, $r_1 = f - Ax_1, \ldots, r_k = f - Ax_k$, when $r_k = o$ then x_k is the solution of $Ax = f$.

Some comments on the above algorithm are in order here.

Remark 1. The conjugate gradient algorithm requires only one matrix vector multiplication, namely Ap_i, in each iterative cycle and it is therefore reasonable to compute the residual vector r_i recursively rather than from $r_i = f - Ax_i$, which requires the additional matrix vector product Ax_i. In fact the algorithm does not require A explicitly but only Ap_i which can often be systematically constructed from minimal data. Because matrix A itself is not needed in the conjugate gradient algorithm, sparseness is more efficiently taken advantage of than in direct solution procedures based on Gauss elimination.

The conjugate gradient algorithm needs only four vectors x_i, p_i, r_i, and Ap_i for its execution, and if the procedure to form Ap_i is concise, then the storage requirement is much lower than that of the Gauss algorithm for the solution of the same $Ax = f$.

Moreover, the huge finite element global stiffness matrix K consists of the assembly of simple small element stiffness matrices, which allow Kp to be computed elementwise in parallel.

Remark 2. An initial guess of the form $x_0 = \alpha x$ does not result in a one-step conjugate gradient solution of $Ax = f$, and it is therefore advisable to scale x_0. Let x_0' be arbitrary

and write $x_0 = \alpha x_0',\ r_0 = f - \alpha A x_0' = f - \alpha q_0$. Then

$$r_0^T r_0 = f^T f - 2\alpha f^T q_0 + \alpha^2 q_0^T q_0 \tag{7.25}$$

and the minimum of $r_0^T r_0$ with respect to α occurs at $\alpha = f^T q_0 / q_0^T q_0$. Now if $x_0' = \alpha x$, then $x_0 = x$.

In the finite element method a good initial guess can be furnished by a low-order discretization for later use in a higher-order, more accurate, approximation.

Remark 3. Diagonal system $Dx = f,\ D_{ii} \neq D_{jj}$, requires n conjugate gradient steps, but the *prescaling* $x = D^{-\frac{1}{2}} x'$ which results in

$$D^{-\frac{1}{2}} D D^{-\frac{1}{2}} x' = I x' = D^{-\frac{1}{2}} f \tag{7.26}$$

creates a system that is solved in one step. It is found that scaling by diagonal matrix D which transforms $Ax = f$ into $D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x' = D^{-\frac{1}{2}} f$ such that $(D^{-\frac{1}{2}} A D^{-\frac{1}{2}})_{ii} = 1$ is generally worth doing before entering the conjugate gradient algorithm.

More complicated prescaling schemes have been suggested to accelerate convergence, but they mar the pristine simplicity of the algorithm.

Remark 4. Let V^m be a subspace of R^n spanned by m eigenvectors of A. If $f \in V^m$ and we start the conjugate gradient algorithm with $x_0 = o,\ r_0 = p_0 = f$, then all generated r_i are in V^m and termination occurs in no more than m steps. This means that, at least in theory, the conjugate gradient algorithm can be applied to the solution of $Ax = f$ with a positive *semi* definite A provided that f is orthogonal to the nullspace of A. For then $Ax = f$ is consistent and $r_0 = p_0 = Ax_0 = f$ is in V^m, the space spanned by eigenvectors corresponding to the nonzero eigenvalues of A.

Consider for example $Ax = f$ with

$$A = \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} \quad \text{and} \quad f = \begin{bmatrix} -1 \\ \\ \\ \\ 1 \end{bmatrix}. \tag{7.27}$$

Matrix A is positive semidefinite with eigenvector $v_1 = [1\ 1\ 1\ 1\ 1]^T$ corresponding to $\lambda_1 = 0$, and $f^T v_1 = 0$. We verify that $x' = [-2\ \ -1\ 0\ 1\ 2]^T$ is a particular solution of $Ax = f$ and its general solution is therefore $x = x' + \alpha v_1$, for arbitrary scalar α.

Starting with $x_0 = [1\ 1\ 1\ 1\ 1]^T$ we compute

$$p_0 = r_0 = f = \begin{bmatrix} -1 \\ \\ \\ \\ 1 \end{bmatrix}\ ,\ Ap_0 = \begin{bmatrix} -1 \\ 1 \\ \\ -1 \\ 1 \end{bmatrix}\ ,\ \begin{matrix} r_0^T r_0 = 2 \\ \\ p_0^T A p_0 = 2 \\ \\ \alpha_0 = 1 \end{matrix}\ ,\ x_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 2 \end{bmatrix}$$

$$r_1 = \begin{bmatrix} -1 \\ \\ \\ \\ 1 \end{bmatrix} - \begin{bmatrix} -1 \\ 1 \\ \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ \\ \\ 1 \\ \\ \end{bmatrix}\ ,\ \begin{matrix} r_1^T r_1 = 2 \\ \\ r_0^T r_0 = 2 \\ \\ \beta_0 = 1 \end{matrix}\ ,\ p_1 = \begin{bmatrix} -1 \\ \\ \\ 1 \\ \\ \end{bmatrix} + \begin{bmatrix} -1 \\ \\ \\ \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ \\ 1 \\ 1 \end{bmatrix}$$

$$Ap_1 = \begin{bmatrix} -1 \\ \\ \\ 1 \\ \\ \end{bmatrix}\ ,\ \begin{matrix} p_1^T A p_1 = 2 \\ \\ r_1^T r_1 = 2 \\ \\ \alpha_1 = 1 \end{matrix}\ ,\ x_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \\ \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}\ ,\ r_2 = \begin{bmatrix} -1 \\ \\ \\ 1 \\ \\ \end{bmatrix} - \begin{bmatrix} -1 \\ \\ \\ \\ 1 \end{bmatrix} = o. \tag{7.28}$$

Another initial guess x_0 would have produced another solution.

The reality of floating-point computations and round-offs is different and often unpleasant. A simple example will obviate to us the treacherous effect round-off errors can have on the workings of the otherwise theoretically pure and perfect conjugate gradient algorithm.

Matrix

$$A = \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & -3 & 1 & & \\ -3 & 6 & -4 & 1 & \\ 1 & -4 & 6 & -4 & 1 \\ & 1 & -4 & 6 & -3 \\ & & 1 & -3 & 2 \end{bmatrix} \tag{7.29}$$

is positive semidefinite with a one-dimensional nullspace spanned by $v_1 = [1\ 1\ \ldots\ 1]^T$. We propose to compute v_1 as the solution of $Ax = o$ using the conjugate gradient algorithm with $x_0 = e_1$, for $A = A(10 \times 10)$. Table 7.1 lists the results of this numerical experiment for single precision computation (machine round-off unit $0.5\,10^{-6}$), while table 7.2 lists the results for double precision computation (machine round-off unit 10^{-16}). In these tables j denotes the step index and ϕ_j^o the angle between r_j and r_0. Theoretically, r_j is in the 9-dimensional space spanned by the nine eigenvectors corresponding to the nine nonzero eigenvalues of A, and with exact arithmetic we should have $r_9 = o$. Not in reality.

In table 7.1 orthogonality of the r vectors is seen to falter at $j = 8$ and r_9 is far from zero. Interestingly enough, if the iterative procedure is continued without interruption the residual vector keeps decreasing and a good approximation to v_1 eventually emerges. The actual behavior of the conjugate gradients method observed in table 7.1 is alarming, suggesting that the method is unstable and of little practical use, but we shall see in a wider range of numerical experiments that for certain matrices possessing special properties, the method of conjugate gradients can be a serious competitor to Gauss elimination. Table 7.1 shows also how round-off errors cause the recursive residual r_j to deviate from the true residual $-Ax_j$. Any error analysis based on the magnitude of the residual must be done with the true residual rather than the recursive one.

Higher computational accuracy makes the algorithm behave in a manner closer to the theoretical. In table 7.2 $\|r_8\|/\|x_8\| = 10^{-2}$ but the next step brings the plunge to $\|r_9\|/\|x_9\| = 10^{-9}$, and a very good approximation to v_1. Continuation of the algorithm further improves the accuracy.

Table 7.1: single precision computation

j	ϕ_j^o	$\log_{10}$ $\|r_j\|/\|x_j\|$	$\log_{10}$ $\|Ax_j\|/\|x_j\|$	$\log_{10}$ $\|\frac{1}{\|x_j\|}x_j - v_1\|$
1	90.00	0.227	0.227	0.050
2	90.00	0.042	0.042	0.038
3	90.00	-0.161	-0.161	0.028
4	90.00	-0.292	-0.292	0.020
5	90.00	-0.410	-0.410	0.012
6	90.00	-0.650	-0.650	0.003
7	90.00	-1.134	-1.134	-0.013
8	89.63	-2.118	-2.119	-0.058
9	95.79	-1.152	-1.152	-0.071
10	51.52	-1.024	-1.024	-1.13
11	104.9	-1.675	-1.675	-2.20
12	90.37	-2.146	-2.146	-2.83
13	44.15	-2.650	-2.650	-3.38
14	81.42	-3.952	-3.956	-4.08
15	96.12	-4.529	-4.537	-4.36
16	74.27	-4.266	-4.253	-4.46
17	83.14	-5.772	-5.085	-4.66
18	115.7	-5.326	-5.311	-4.67
19	68.06	-5.916	-5.125	-4.73

Table 7.2: double precision computation

j	ϕ_j^o	$\log_{10}$ $\|r_j\|/\|x_j\|$	$\log_{10}$ $\|Ax_j\|/\|x_j\|$	$\log_{10}$ $\|\frac{x_j}{\|x_j\|} - v_1\|$
1	90.00	0.227	0.227	0.050
2	90.00	0.042	0.042	0.038
3	90.00	-0.161	-0.161	0.028
4	90.00	-0.292	-0.292	0.020
5	90.00	-0.410	-0.410	0.012
6	90.00	-0.650	-0.650	0.003
7	90.00	-1.134	-1.134	-0.013
8	90.00	-2.121	-2.121	-0.059
9	102.22	-9.075	-9.075	-10.26
10	64.92	-10.71	-10.71	-11.75
11	93.58	-11.52	-11.52	-12.16
12	39.75	-11.92	-11.92	-12.80
13	72.35	-12.94	-12.93	-13.50
14	104.44	-13.59	-13.59	-13.64
15	80.23	-14.08	-14.06	-13.88
16	90.80	-15.28	-14.84	-14.17
17	89.10	-14.92	-14.77	-14.00
18	95.05	-13.13	-13.13	-11.13
19	73.87	-12.63	-12.63	-11.13

exercises

7.3.1. Repeat the computation described in table 7.1. Terminate the iterative process after n steps and use the last approximation as a new initial guess.

7.4 Variational interpretation

The conjugate gradient algorithm of the previous section for the iterative solution of the linear system $Ax = f$ with a positive definite and symmetric A can be interpreted as a procedure for the minimization of the quadratic functional

$$\phi(x) = \frac{1}{2}x^T Ax - x^T f \tag{7.30}$$

to locate its unique minimum $\phi(s)$ at $x = s,\ As = f$.

By the conjugate gradient method $x_{i+1} = x_i + \alpha_i p_i$ and therefore

$$\phi(x_{i+1}) = \phi(x_i + \alpha_i p_i) = \phi(x_i) + \frac{1}{2}\alpha_i^2 p_i^T Ap_i - \alpha_i p_i^T r_i. \tag{7.31}$$

The minimum of $f(x_{i+1})$ with respect to α_i occurs at

$$\alpha_i p_i^T A p_i - p_i^T r_i = 0 \tag{7.32}$$

or $\alpha_i = p_i^T r_i / p_i^T A p_i$, the same α_i as in Theorem 7.1. With this α_i,

$$\phi(x_{i+1}) - \phi(x_i) = -\frac{1}{2}\frac{(r_i^T r_i)^2}{p_i^T A p_i} \tag{7.33}$$

and the conjugate gradient algorithm *monotonically decreases* $\phi(x)$. In this sense the conjugate gradient algorithm is a true iterative method; each step brings us closer to the solution.

In fact the minimization of $\phi(x_{i+1})$ is carried out not only in the direction of p_i but in the whole subspace of $p_0, p_1, \ldots, p_i$; and in this sense the search for the minimum is optimal.

Theorem 7.10. *The conjugate gradient algorithm is optimal.*

Proof. From

$$x_{i+1} = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_i p_i \tag{7.34}$$

it results that

$$\begin{aligned}\phi(x_{i+1}) = \phi(x_0) + \frac{1}{2}\alpha_0^2 p_0^T A p_0 + \frac{1}{2}\alpha_1^2 p_1^T A p_1 + \cdots + \frac{1}{2}\alpha_i^2 p_i^T A p_i \\ -\alpha_0 p_0^T r_0 - \alpha_1 p_1^T r_1 - \cdots - \alpha_i p_i^T r_i.\end{aligned} \tag{7.35}$$

The best α's are obtained from

$$\frac{\partial \phi_{i+1}}{\partial \alpha_0} = \frac{\partial \phi_{i+1}}{\partial \alpha_1} = \cdots = \frac{\partial \phi_{i+1}}{\partial \alpha_i} = 0 \tag{7.36}$$

or

$$\begin{aligned}\alpha_0 p_0^T A p_0 - p_0^T r_0 &= 0 \\ \alpha_1 p_1^T A p_1 - p_1^T r_0 &= 0 \\ &\vdots \\ \alpha_i p_i^T A p_i - p_i^T r_0 &= 0.\end{aligned} \tag{7.37}$$

Since $p_i^T r_0 = p_i^T r_i$, the α's are precisely those of the conjugate gradient method and $x_{i+1} = x_i + \alpha_i p_i$. End of proof.

Each step of the conjugate gradient algorithm can be construed as a two-dimensional minimization of $\phi(x)$ to select the optimal α_{i+1} and β_i. Indeed, if we write

$$\begin{aligned} x_{i+1} = x_i + \alpha_i p_i &= x_i + \alpha_i(r_i + \beta_{i-1}p_{i-1}) \\ &= x_i + \alpha_i r_i + (\alpha_i\beta_{i-1})p_{i-1} \end{aligned} \tag{7.38}$$

and minimize $\phi(x_{i+1})$ with respect to both α_i and β_{i-1} we obtain the conjugate gradients' expressions for the two scalars.

A geometrical interpretation of the two-dimensional conjugate gradient minimization is shown in Fig. 7.1.

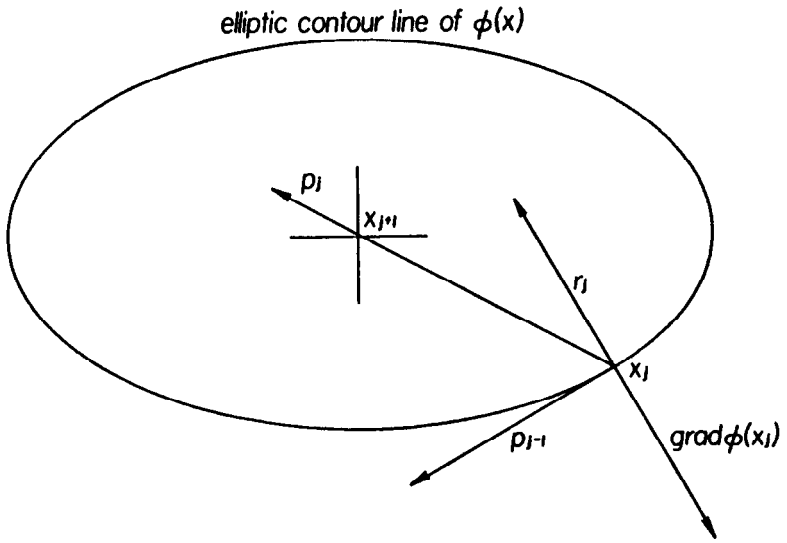

Fig. 7.1

The length of residual vector r_i is not reduced monotonically by the conjugate gradient algorithm, but

Theorem 7.11. *The conjugate gradient approximations $x_0, x_1, \ldots, x_i$ are distinct, and*

$$\phi(x_{i+1}) - \phi(x_i) = -\frac{1}{2}\alpha_i\|r_i\| < 0 \tag{7.39}$$

so that $\phi(x_{i+1}) < \phi(x_i)$.

Proof. Since

$$\|p_i\|^2 = \|r_i\|^4\Big(\|r_0\|^{-2} + \|r_1\|^{-2} + \cdots + \|r_i\|^{-2}\Big) \tag{7.40}$$

$p_i \neq o$ if $r_0 \neq o,\ r_1 \neq o, \ldots, r_i \neq o$, and

$$\|x_{i+1} - x_i\| = \alpha_i \|p_i\| = \frac{\|r_i\|^2 \|p_i\|}{p_i^T A p_i} \neq 0. \tag{7.41}$$

From $x_{i+1} = x_i + \alpha_i p_i$ we have that

$$\phi(x_{i+1}) - \phi(x_i) = \frac{1}{2}\alpha_i^2 p_i^T A p_i - \alpha_i p_i^T r_i \tag{7.42}$$

which with $p_i^T r_i = r_i^T r_i$ and $\alpha_i = r_i^T r_i / p_i^T A p_i$ produces the first inequality in the theorem. End of proof.

The next theorem is more forceful.

Theorem 7.12. *At each step of the conjugate gradient method the error vector $e_i = s - x_i$ is reduced in length. In fact*

$$\|e_{i+1}\|^2 - \|e_i\|^2 = -\frac{p_i^T p_i}{p_i^T A p_i}(e_{i+1}^T A e_{i+1} + e_i^T A e_i). \tag{7.43}$$

Proof. Because the conjugate gradient algorithm terminates in a finite number of steps we may write

$$s = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_m p_m \tag{7.44}$$

which with

$$x_i = x_0 + \alpha_0 p_0 + \cdots + \alpha_{i-1} p_{i-1} \tag{7.45}$$

becomes

$$e_i = \alpha_i p_i + \alpha_{i+1} p_{i+1} + \cdots + \alpha_m p_m. \tag{7.46}$$

Consequently, since $p_i^T A p_j = 0,\ i \neq j$

$$e_i^T A e_i = \alpha_i^2 p_i^T A p_i + \cdots + \alpha_m^2 p_m^T A p_m \tag{7.47}$$

or with $\alpha_i = \|r_i\|^2 / p_i^T A p_i$

$$e_i^T A e_i = \alpha_i \|r_i\|^2 + \cdots + \alpha_m \|r_m\|^2. \tag{7.48}$$

On the other hand, since $x_{i+1} = x_i + \alpha_i p_i$, then

$$-e_{i+1} = -e_i + \alpha_i p_i \tag{7.49}$$

and

$$\|e_{i+1}\|^2 = \|e_i\|^2 + \alpha_i^2\|p_i\|^2 - 2\alpha_i p_i^T e_i \tag{7.50}$$

or, by eq. (7.46)

$$\|e_{i+1}\|^2 = \|e_i\|^2 - \alpha_i^2\|p_i\|^2 - 2\alpha_i(\alpha_{i+1}p_i^T p_{i+1} + \cdots + \alpha_m p_i^T p_m) \tag{7.51}$$

which with

$$p_i^T p_j = \|r_j\|^2\|p_i\|^2/\|r_i\|^2 \qquad i \le j \tag{7.52}$$

becomes

$$\|e_{i+1}\|^2 = \|e_i\|^2 - \frac{p_i^T p_i}{p_i^T A p_i}(\alpha_i\|r_i\|^2 + 2\alpha_{i+1}\|r_{i+1}\|^2 + \cdots + 2\alpha_m\|r_m\|^2) \tag{7.53}$$

or

$$\|e_{i+1}\|^2 = \|e_i\|^2 - \frac{p_i^T p_i}{p_i^T A p_i}(\alpha_i^2 p_i^T A p_i + 2\alpha_{i+1}^2 p_{i+1}^T A p_{i+1} + \cdots + 2\alpha_m^2 p_m^T A p_m) \tag{7.54}$$

from which, using eq. 7.46, the equality of the theorem readily results. End of proof.

exercises

7.4.1. Consider the iterative method whereby $x_1 = x_0 + \alpha_0 r_0$, $r_0 = f - Ax_0$. Determine α_0 by the condition that $\rho_1^2(\alpha_0) = r_1^T r_1$ is minimal with respect to α_0.

7.5 Practicalities

How does the conjugate gradient algorithm compare with Gauss elimination for the solution of the linear system $Ax = f$ with a positive definite and symmetric A? In two very fundamental ways Gauss elimination is superior: the arithmetical work it requires to solve the system is a known function of such explicit parameters as bandwidth and number of equations, and is therefore predictable; and it is numerically stable or insensitive to round-off errors. Vast theoretical and numerical experience suggests that the round-off errors incurred during the Gauss solution of the linear system is comparable with the round-off error suffered by the system just by storage.

The drawbacks of Gauss elimination are that its execution requires the coefficient matrix to be in tabular form, necessitating complicated and costly programming expedients to make

it efficient in the presence of sparseness, and it is therefore not well-suited to take advantage of the highly structured matrices of mathematical physics that are often compactly described by a formula or an algorithm.

Generally, Gauss elimination is also faster. The number of steps needed with conjugate gradients depends, as we shall soon see, on the intrinsic nature of A and the accuracy of the arithmetic, and cannot be accurately predicted. On the rough assumption that n conjugate gradient steps are needed to solve an $n \times n$ system of linear equations of half bandwidth k, some $2kn^2$ arithmetical operations are required by the algorithm, while Gauss elimination completes the same solution in only $\frac{1}{2}k^2n$ operations.

Storagewise the conjugate gradient algorithm has the marked advantage over Gauss elimination in that it does not require matrix A itself but only the product Ap, and this matrix vector multiplication can be done concisely and efficiently, avoiding sparseness zero operations. Repetitiveness and composition, common in finite difference and finite element matrices, is fully exploited when forming Ap and the storage requirements for A can be minimal. The algorithm is simple to program and exacts little overhead. Symmetries manifested in repeated eigenvalues are naturally taken advantage of by the algorithm in reducing the number of iterations for a solution.

But the decisive advantage of the conjugate gradients algorithm–its true salvation–comes from its potent iterative nature and its fast convergence in certain recognized circumstances. A theoretical analysis of the algorithm deeper than that of the previous four sections is too copious for this book, but we should be able to form a balanced opinion on the workings, practicalities, failings, and worth of the algorithm as a practical tool for the solution of the large systems of linear equations of mathematical physics by looking at some carefully designed numerical experiments.

Symmetric matrix A is described by its n eigenvalues and the corresponding n orthogonal eigenvectors. In Chapter 3 we considered the finite difference matrices resulting from the discretization of the one-dimensional, second- and fourth-order equations– $d^2u/dx^2 = f(x)$ and $d^4u/dx^4 = f(x)$ with various physically plausible boundary conditions. Stiffness matrix $A = A(n \times n)$ for the one-dimensional second-order boundary value problem is with an

eigenvalue distribution close to

$$\lambda_i = \lambda_1 i^2 \ , \quad i = 1, 2, \ldots, n \tag{7.55}$$

and that of the fourth-order problem with eigenvalue distribution close to

$$\lambda_i = \lambda_1 i^4 \ , \quad i = 1, 2, \ldots, n. \tag{7.56}$$

We start our numerical experiments with an examination of the performance of the conjugate gradient algorithm on systems of equations with such matrices, assuming that A is the diagonal $D_{ii} = \lambda_i = i^\alpha$ with α ranging from 0.1 to 4. As starting vectors we choose $p_0 = r_0 = [1\ 1\ \ldots\ 1]^T$ and compute the error $e_j = s - x_j$ at step j from $s - x_j = A^{-1} r_j$.

Figure 7.2 shows the reduction of $\|e_j\| = \sqrt{e_j^T e_j}$ as iteration proceeds through its cycles. Computation described in this figure is done in single precision (round-off error unit $u = 0.5\ 10^{-6}$) for a 20×20 system with a coefficient matrix having the eigenvalues $\lambda_i = i^\alpha$. The two most striking features of Fig. 7.2 is that the conjugate gradient algorithm is capable of producing good approximate solutions in fewer than n steps even for matrices with distinct eigenvalues if they are clustered; and that round-off errors can have a marked effect in delaying convergence in ill-conditioned circumstances.

For $\alpha = 0.1$

$$\lambda_1 = 1., \ \lambda_2 = 1.0718, \ldots, \lambda_{19} = 1.3424, \ \lambda_{20} = 1.3493 \tag{7.57}$$

and convergence is nearly *linear* and of such impressive rate that $\|e_j\|$ gets to be 10^{-8} in less than $n/3$ steps. As α in $\lambda_i = i^\alpha$ is increased convergence slows down and for $\alpha = 2, \ n$ steps bring a mere $\|e_j\| = 10^{-1}$. Continuation of the iteration beyond n cycles brings after a few additional steps a sudden drop in the error. Convergence after more than n steps is clearly the insidious work of round-off errors destroying orthogonalities and A-orthogonalities. This phenomenon becomes more pronounced for $\alpha = 4$. Here even extra n steps barely reduce the error, but a few more steps bring a nearly vertical decline in $\|e_j\|$.

Higher, or extended precision (round-off unit $u = 10^{-32}$) computations have a decisive effect only after n steps as seen in Fig. 7.3. For $\alpha \leq 1$ use of this extreme (and costly) accuracy does not seem to alter the convergence pattern of the algorithm. For $\alpha > 1$

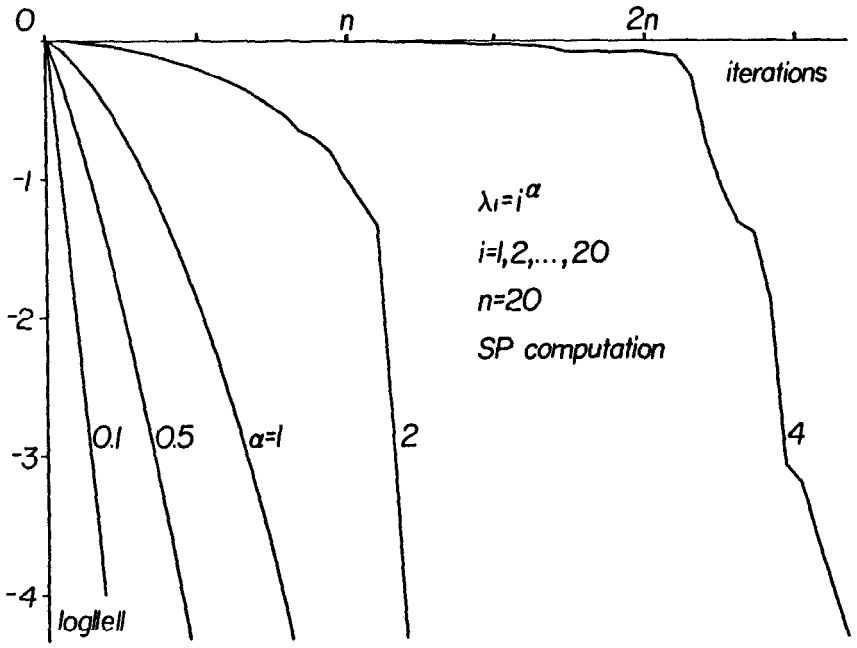

Fig. 7.2

convergence is still nearly the same for single and extended precision computations, but termination occurs here invariably at n steps. Slow convergence is due to the nature of the matrix, and round-off errors only stretch it out beyond n steps.

Fast convergence of the conjugate gradient algorithm in cases of matrices spectrally close to the identity implies that accurate matrix inversion is wasteful in such cases, and in this lies the hope of the algorithm.

The conjugate gradient algorithm is of little practical interest for the small finite difference matrices that arise in the discretization of one-dimensional boundary value problems. It is the large multi-dimensional discretizations we are after.

From previous experience we have an indication that the theoretical convergence of the conjugate gradient algorithm depends on the eigenvalue spread of matrix A, that is on λ_n/λ_1, as well as on their distribution. To separate the two effects we experiment next with a diagonal matrix A with eigenvalues recursively given by

$$\lambda_1 = \lambda_1 \ , \ \lambda_{i+1} = \lambda_i + \lambda i^\alpha \quad i = 1, 2, \ldots, n-1 \tag{7.58}$$

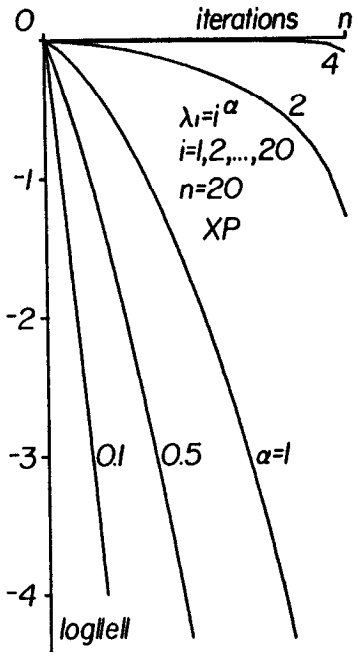

Fig. 7.3

in which λ and α are two parameters. We have for this spectrum that

$$\lambda_n = \lambda_1 + \lambda(1^\alpha + 2^\alpha + \cdots + (n-1)^\alpha) \tag{7.59}$$

and for any chosen λ_1 and α we fix λ so that $\lambda_n = 1$ and $\lambda_n/\lambda_1 = \lambda_1^{-1}$. In this way λ_1 determines the spread of the eigenvalues and α their distribution. Figure 7.4 shows this spectrum for $\alpha = 0,\ \alpha = 1,\ \alpha = 2$ and $\alpha = 4$ for the fixed values of $n = 100$ and $\lambda_1 = 10^{-4}$. The choice $\alpha = 0$ produces a uniform eigenvalue distribution while an increase in α causes the eigenvalues to crowd near λ_1.

Figures 7.5 to 7.9 are for α that grows from 0 to 2, and in each figure $\kappa = \lambda_n/\lambda_1$ is varied between 10^1 and 10^6. Computation of the results in Figs. 7.5 to 7.9 are done in double precision ($u = 10^{-16}$). Ill-conditioned matrices, say $k > 3$, are invariably associated with slow initial convergence, but if the eigenvalues are nearly uniformly distributed, that is, α is near zero, then a clear turning point exists at which convergence suddenly accelerates to bring about an early termination. Such a turning point disappears as α is increased, and convergence becomes almost perfectly linear with a constant rate ρ so that

$$\|e_j\| = \rho^j \|e_0\|. \tag{7.60}$$

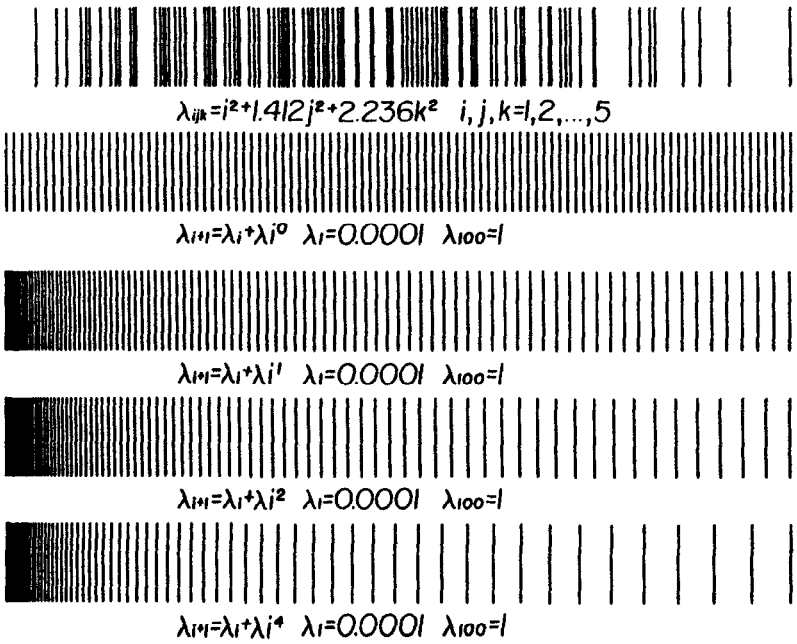

Fig. 7.4

But it can be painfully slow. Round-off errors are implicated in continuation beyond n steps. Careful numerical analysis establishes that the linear rate of convergence ρ of the conjugate gradient algorithm is given by

$$\rho = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \tag{7.61}$$

where $\kappa = \lambda_n / \lambda_1$ is the spectral condition number of matrix A.

Hence the method of conjugate gradients is most suited for large well-conditioned systems. There it can rival Gauss elimination. Such systems *are* common in applied linear algebra.

Finite difference and finite element matrices arising from the approximation of omnipresent second-order spatial differential equation

$$u_{xx} + u_{yy} + u_{zz} + f(x, y, z) = 0 \tag{7.62}$$

with appropriate boundary conditions have a spectrum typically given by

$$\lambda_{ijk} = \alpha i^2 + \beta j^2 + \gamma k^2 \quad i, j, k = 1, 2, \ldots, m \tag{7.63}$$

with $\alpha, \beta, \gamma > 0$. The top of Fig. 7.4 shows the spectrum for $\alpha = 1.0$, $\beta = 1.412$, $\gamma = 2.236$, and $m = 5 (n = 5 \times 5 \times 5 = 125)$.

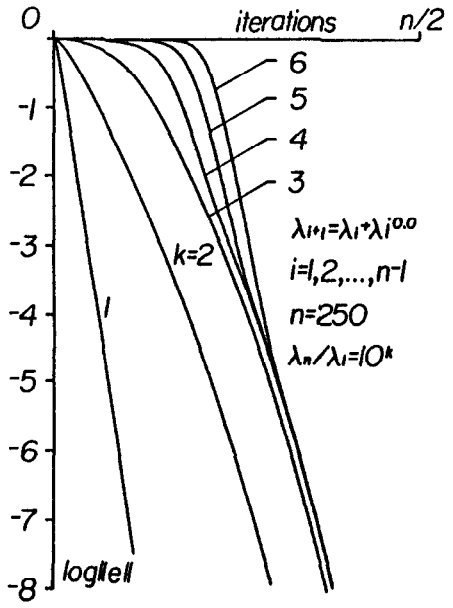

Fig. 7.5

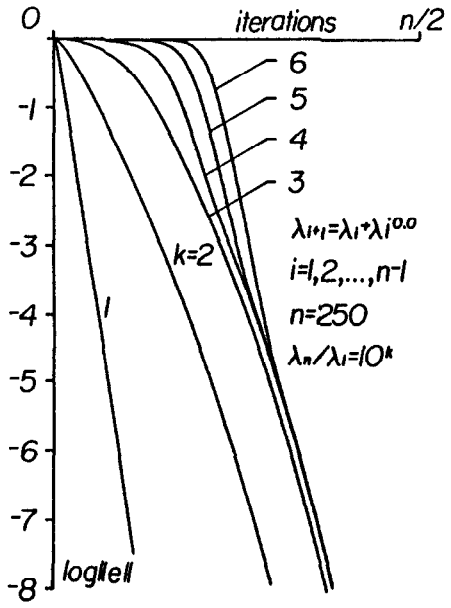

Fig. 7.6

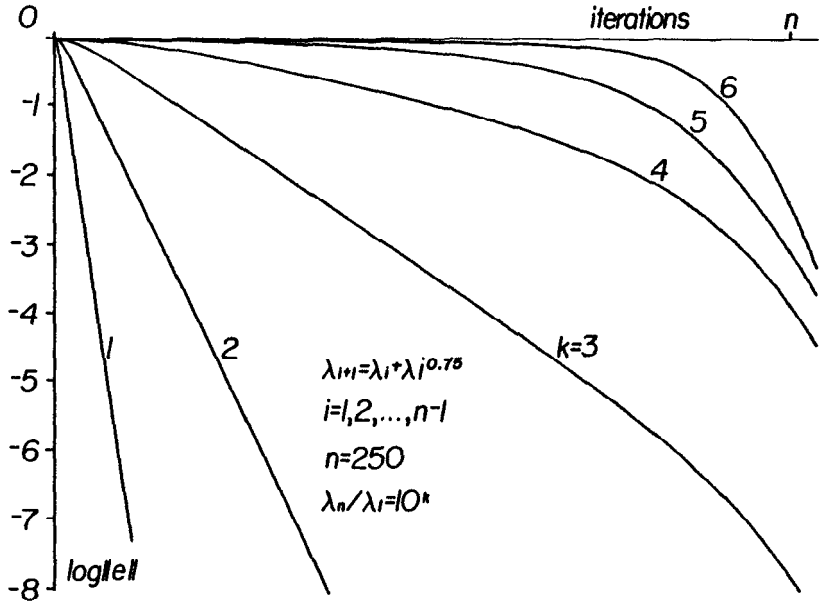

Fig. 7.7

Figure 7.10 shows the convergence of the conjugate gradients algorithm for a 900×900 system simulating a two-dimensional eq. (7.62). Iteration is started with the initial guess $p_0 = r_0 = \alpha[1\ 1 \ldots 1]^T$, where α is adjusted so that $\|e_0\| = 1$. Extended precision ($u = 10^{-32}$)

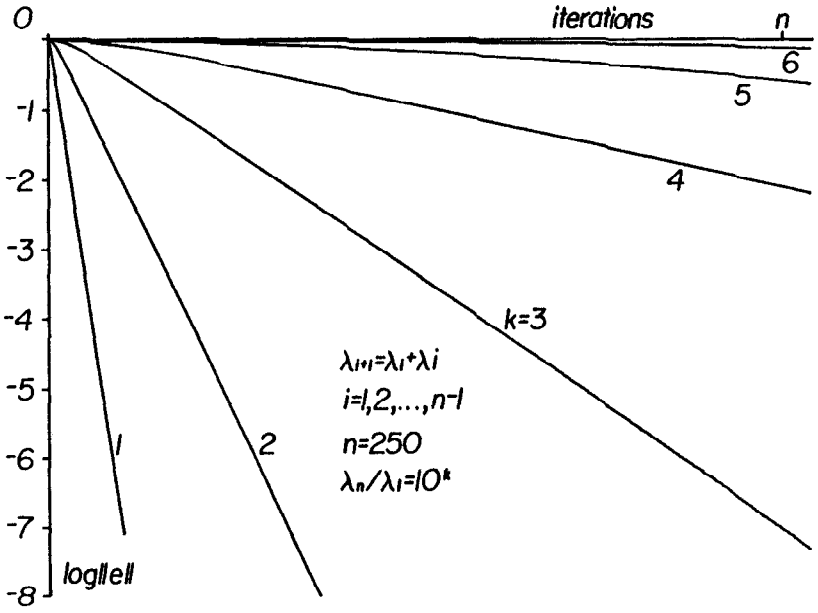

Fig. 7.8

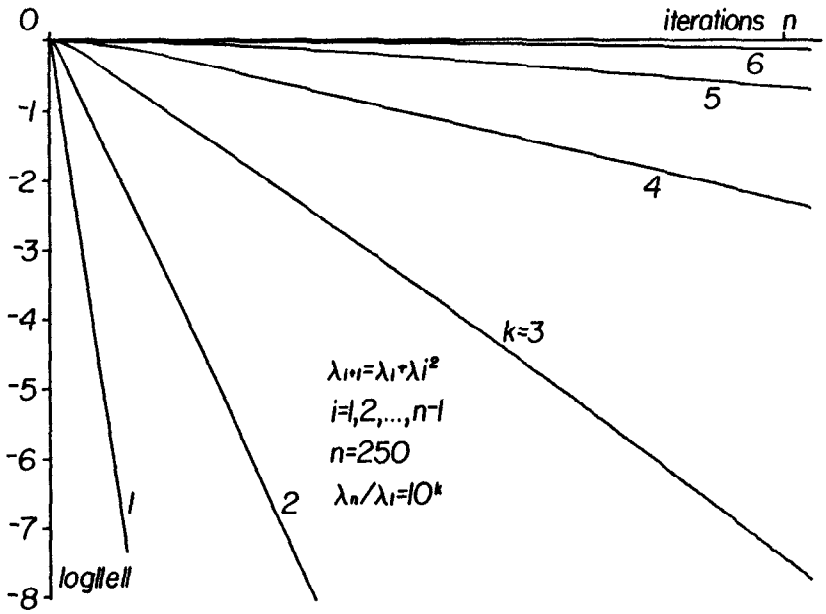

Fig. 7.9

computation is seen to have a slight advantage over single precision computation. Some 200 iterations reduce $\|e_j\|$ to 10^{-8}. Performance of the conjugate gradient algorithm becomes more impressive in three dimensions. Figure 7.11 is for an 8000×8000 system for which $\|e_j\|$ is reduced from 1 to 10^{-8} in no more than 160 steps, in agreement with the prediction

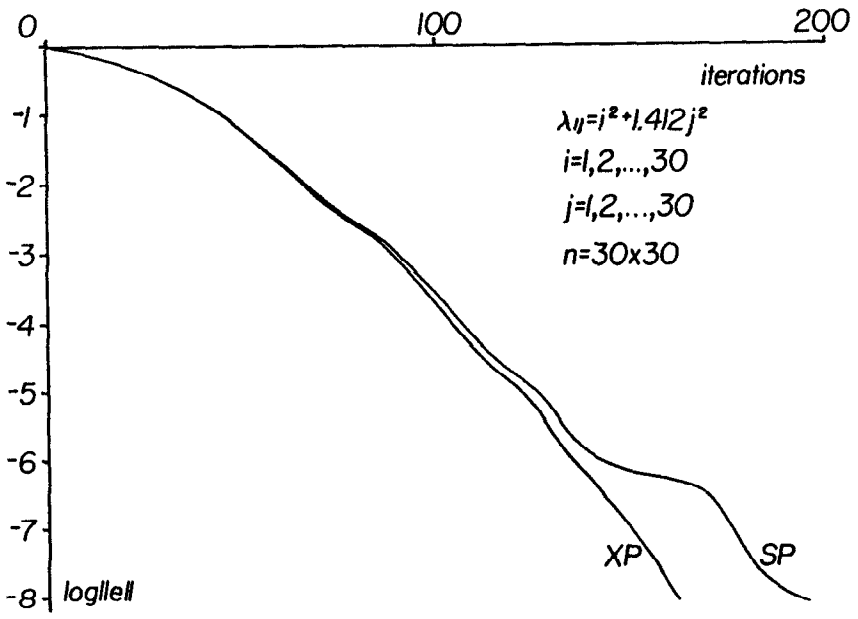

Fig. 7.10

of eq. 7.61. Here, $\kappa = \lambda_n/\lambda_1 = m^2$ and the algorithm's rate of convergence is

$$\rho = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} = \frac{m-1}{m+1} \ . \tag{7.64}$$

To achieve an error reduction from 1 to $\|e_j\| = 10^{-7}$ the j steps needed are given by

$$10^{-7} = (\frac{m-1}{m+1})^j \quad \text{or} \quad -7 = j \log \frac{m-1}{m+1} \tag{7.65}$$

which for $m = 20$ yields $j = 161$. For large m the above formula means that *the number of conjugate gradient steps needed to achieve an acceptable accuracy is proportional to m only.*

Figure 7.12 shows the convergence of the conjugate gradients algorithm for a system with a matrix that has the eigenvalues

$$\lambda_{ij} = i^4 + 1.412\ j^4 \quad i = 1, 2, \ldots, 20, \quad j = 1, 2, \ldots, 20 \ . \tag{7.66}$$

Such matrices occur in the discretization of the *biharmonic* or thin plate bending equation

$$\frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4} = f(x, y) \tag{7.67}$$

with appropriate boundary conditions. Convergence is slower here and the round-off error effects more disruptive.

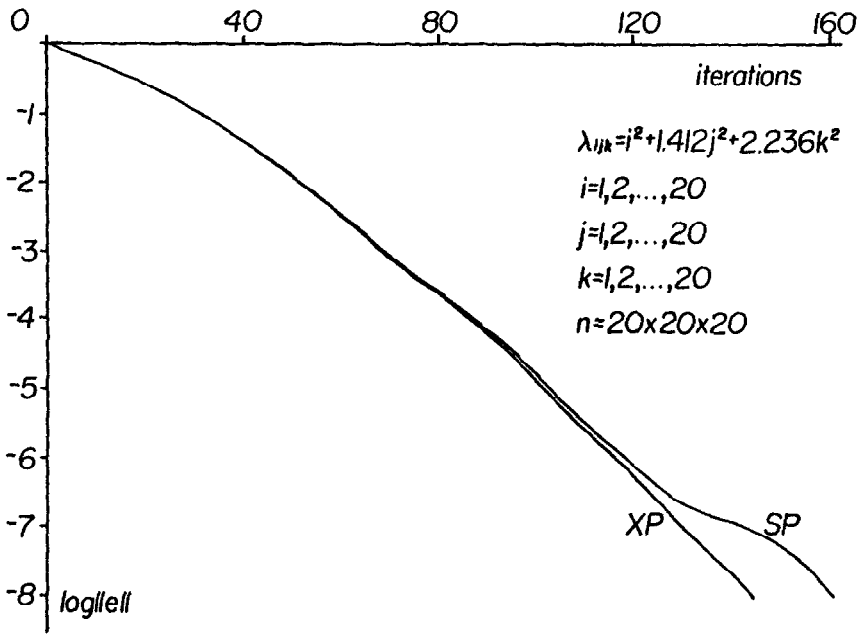

Fig. 7.11

Timely termination of the iterative process is obviously important. Computation of the *true* residual $r_j = f - Ax_j$ and termination when $\|r_j\|/\|f\|$ reaches the level of the round-off unit u, or when $\|r_j\|$ ceases to change, is reasonable but expensive since it requires the computation of Ax_i at each step at double the cost. Figure 7.13 shows the magnitude reductions of the true, or exact, residual r_e and the computed or recursive residual r_c. Computation is done in single precision ($u = 0.5\ 10^{-6}$) and we notice the existence of a minimal value to which $\|r_e\|$ can reduce, after which it remains constant. In contrast, $\|r_c\|$ declines relentlessly even after $\|r_e\|$ ceased to do so. Figure 7.13 suggests that the algorithm be stopped as soon as $\|r_c\|$ reaches the prevailing round-off error level, here $\|r_c\| = 10^{-6.3}$. Figure 7.14 repeats the computation in double precision ($u = 10^{-16}$) with the suggested termination criterion.

Figures 7.15 and 7.16 are for the more difficult biharmonic case, lending further credence to the suggested termination signal $\|r_c\| = u$.

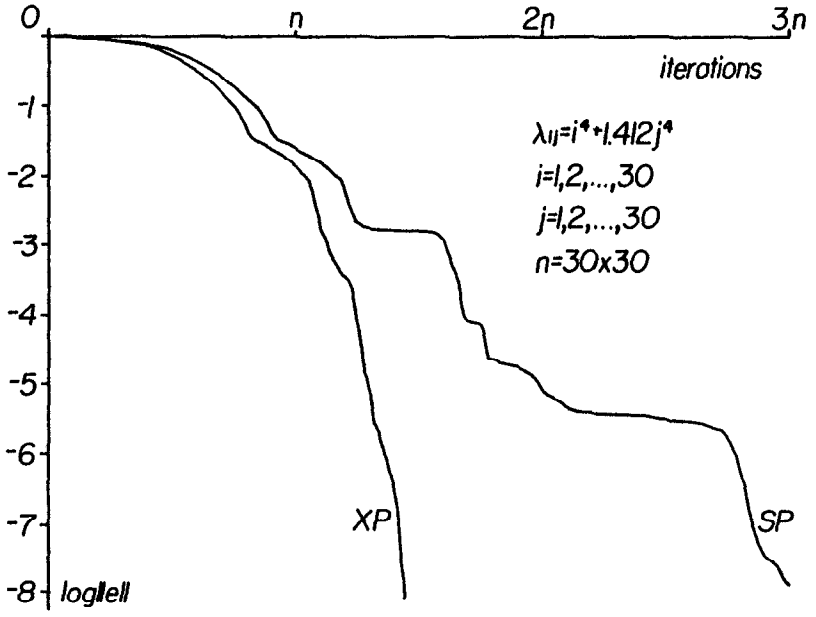

Fig. 7.12

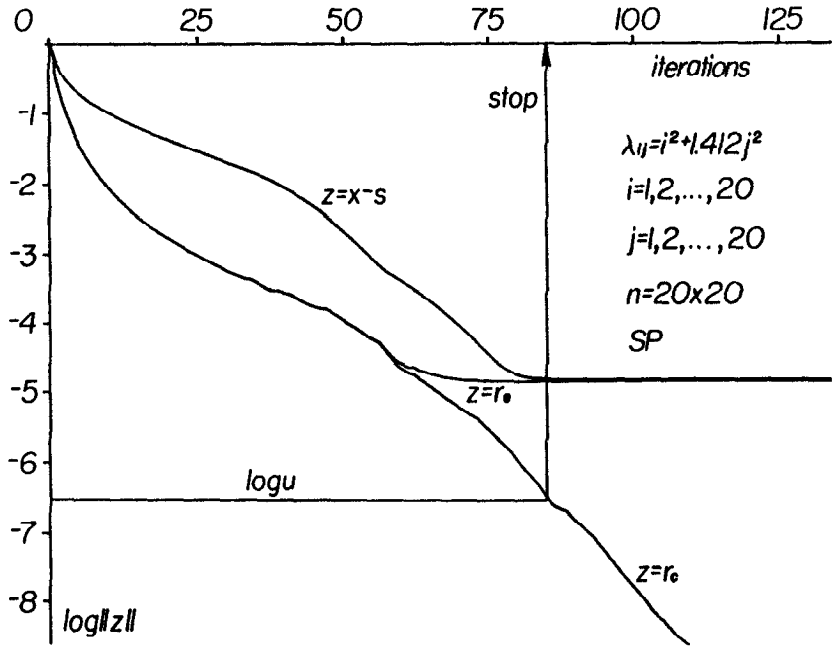

Fig. 7.13

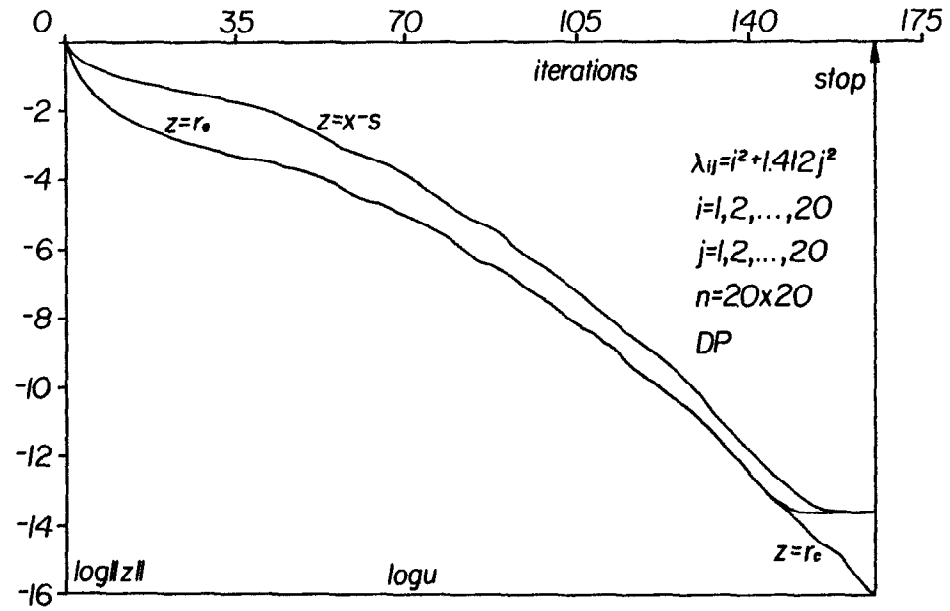

Fig. 7.14

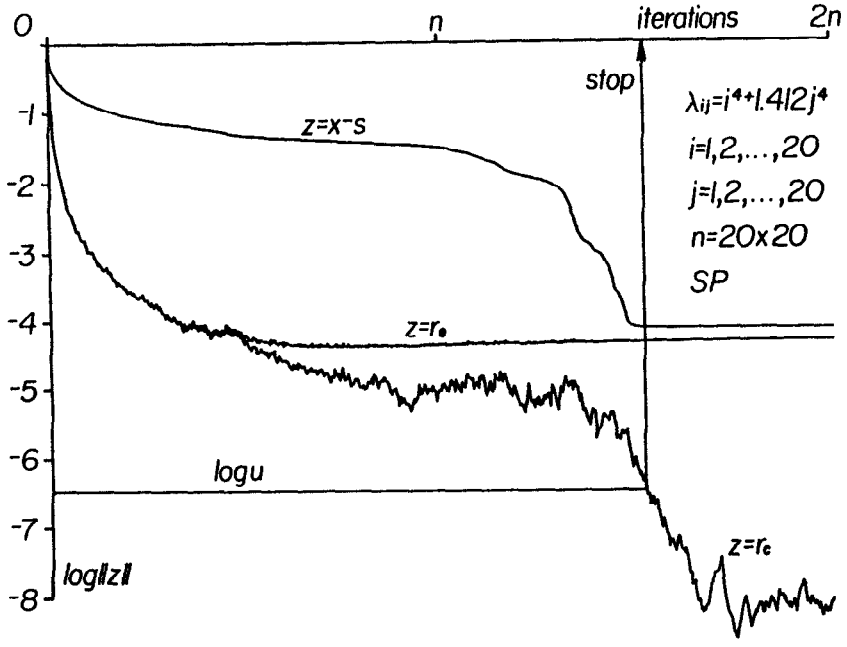

Fig. 7.15

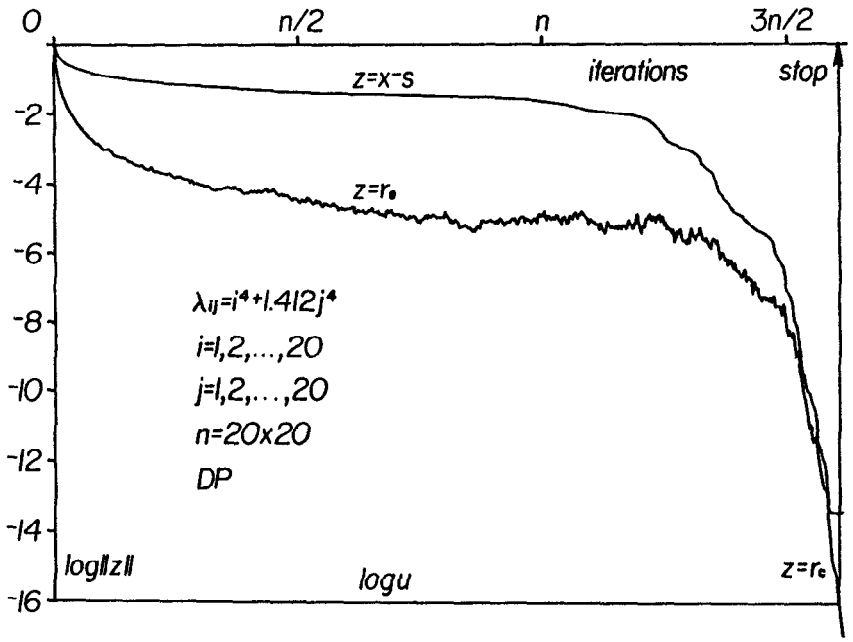

Fig. 7.16

exercises

7.5.1. Second- and fourth-order boundary value problems with the differential equations

$$-\frac{d^2u}{dx^2}+\epsilon u=f(x) \quad \text{and} \quad \frac{d^4u}{dx^4}+\epsilon u=f(x)\ ,\ \epsilon>0$$

are common. Corresponding to the first we have the typical linear finite difference $n\times n$ system

$$(\frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}+\epsilon\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix})x=f,\ h=\frac{1}{n+1}$$

and corresponding to the second the typical

$$(\frac{1}{h^4}\begin{bmatrix} 5 & -4 & 1 & & \\ -4 & 6 & -4 & 1 & \\ 1 & -4 & 6 & -4 & 1 \\ & 1 & -4 & 6 & -4 \\ & & 1 & -4 & 5 \end{bmatrix}+\epsilon\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix})x=f\ ,\ h=\frac{1}{n+1}.$$

Study the influence of ϵ on the performance of the conjugate gradient method.