

CHAPTER 5

Conjugate Gradient Methods

Our interest in conjugate gradient methods is twofold. First, they are among the most useful techniques for solving large linear systems of equations. Second, they can be adapted to solve nonlinear optimization problems. The remarkable properties of both *linear* and *nonlinear* conjugate gradient methods will be described in this chapter.

The *linear* conjugate gradient method was proposed by Hestenes and Stiefel in the 1950s as an iterative method for solving linear systems with positive definite coefficient matrices. It is an alternative to Gaussian elimination that is well suited for solving large problems. The performance of the linear conjugate gradient method is determined by the

distribution of the eigenvalues of the coefficient matrix. By transforming, or *preconditioning*, the linear system, we can make this distribution more favorable and improve the convergence of the method significantly. Preconditioning plays a crucial role in the design of practical conjugate gradient strategies. Our treatment of the linear conjugate gradient method will highlight those properties of the method that are important in optimization.

The first *nonlinear* conjugate gradient method was introduced by Fletcher and Reeves in the 1960s. It is one of the earliest known techniques for solving large-scale nonlinear optimization problems. Over the years, many variants of this original scheme have been proposed, and some are widely used in practice. The key features of these algorithms are that they require no matrix storage and are faster than the steepest descent method.

5.1 THE LINEAR CONJUGATE GRADIENT METHOD

In this section we derive the linear conjugate gradient method and discuss its essential convergence properties. For simplicity, we drop the qualifier “linear” throughout.

The conjugate gradient method is an iterative method for solving a linear system of equations

$$Ax = b, \quad (5.1)$$

where A is an $n \times n$ symmetric positive definite matrix. The problem (5.1) can be stated equivalently as the following minimization problem:

$$\min \phi(x) \stackrel{\text{def}}{=} \frac{1}{2}x^T Ax - b^T x, \quad (5.2)$$

that is, both (5.1) and (5.2) have the same unique solution. This equivalence will allow us to interpret the conjugate gradient method either as an algorithm for solving linear systems or as a technique for minimizing convex quadratic functions. For future reference, we note that the gradient of ϕ equals the residual of the linear system, that is,

$$\nabla\phi(x) = Ax - b \stackrel{\text{def}}{=} r(x), \quad (5.3)$$

so in particular at $x = x_k$ we have

$$r_k = Ax_k - b. \quad (5.4)$$

CONJUGATE DIRECTION METHODS

One of the remarkable properties of the conjugate gradient method is its ability to generate, in a very economical fashion, a set of vectors with a property known as *conjugacy*. A

set of nonzero vectors $\{p_0, p_1, \dots, p_l\}$ is said to be *conjugate* with respect to the symmetric positive definite matrix A if

$$p_i^T A p_j = 0, \quad \text{for all } i \neq j. \quad (5.5)$$

It is easy to show that any set of vectors satisfying this property is also linearly independent. (For a geometrical illustration of conjugate directions see Section 9.4.)

The importance of conjugacy lies in the fact that we can minimize $\phi(\cdot)$ in n steps by successively minimizing it along the individual directions in a conjugate set. To verify this claim, we consider the following *conjugate direction* method. (The distinction between the conjugate gradient method and the conjugate direction method will become clear as we proceed.) Given a starting point $x_0 \in \mathbb{R}^n$ and a set of conjugate directions $\{p_0, p_1, \dots, p_{n-1}\}$, let us generate the sequence $\{x_k\}$ by setting

$$x_{k+1} = x_k + \alpha_k p_k, \quad (5.6)$$

where α_k is the one-dimensional minimizer of the quadratic function $\phi(\cdot)$ along $x_k + \alpha p_k$, given explicitly by

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}; \quad (5.7)$$

see (3.55). We have the following result.

Theorem 5.1.

For any $x_0 \in \mathbb{R}^n$ the sequence $\{x_k\}$ generated by the conjugate direction algorithm (5.6), (5.7) converges to the solution x^* of the linear system (5.1) in at most n steps.

PROOF. Since the directions $\{p_i\}$ are linearly independent, they must span the whole space \mathbb{R}^n . Hence, we can write the difference between x_0 and the solution x^* in the following way:

$$x^* - x_0 = \sigma_0 p_0 + \sigma_1 p_1 + \dots + \sigma_{n-1} p_{n-1},$$

for some choice of scalars σ_k . By premultiplying this expression by $p_k^T A$ and using the conjugacy property (5.5), we obtain

$$\sigma_k = \frac{p_k^T A (x^* - x_0)}{p_k^T A p_k}. \quad (5.8)$$

We now establish the result by showing that these coefficients σ_k coincide with the step lengths α_k generated by the formula (5.7).

If x_k is generated by algorithm (5.6), (5.7), then we have

$$x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_{k-1} p_{k-1}.$$

By premultiplying this expression by $p_k^T A$ and using the conjugacy property, we have that

$$p_k^T A(x_k - x_0) = 0,$$

and therefore

$$p_k^T A(x^* - x_0) = p_k^T A(x^* - x_k) = p_k^T (b - Ax_k) = -p_k^T r_k.$$

By comparing this relation with (5.7) and (5.8), we find that $\sigma_k = \alpha_k$, giving the result. \square

There is a simple interpretation of the properties of conjugate directions. If the matrix A in (5.2) is diagonal, the contours of the function $\phi(\cdot)$ are ellipses whose axes are aligned with the coordinate directions, as illustrated in Figure 5.1. We can find the minimizer of this function by performing one-dimensional minimizations along the coordinate directions

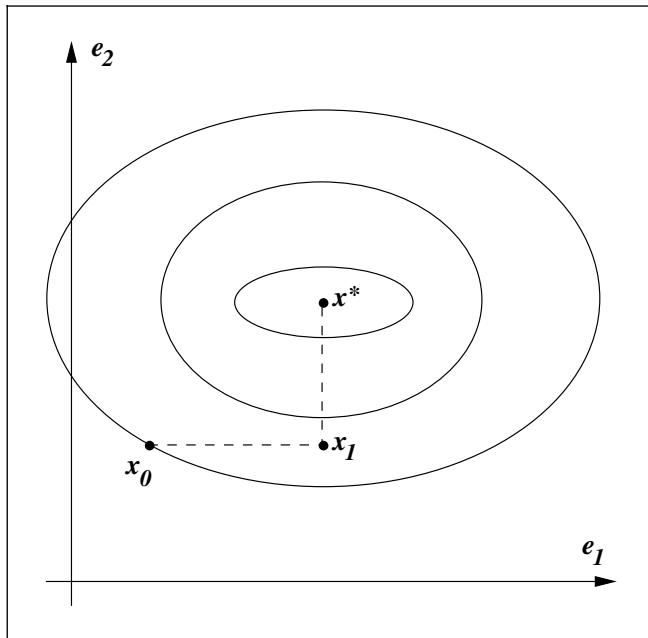


Figure 5.1 Successive minimizations along the coordinate directions find the minimizer of a quadratic with a diagonal Hessian in n iterations.

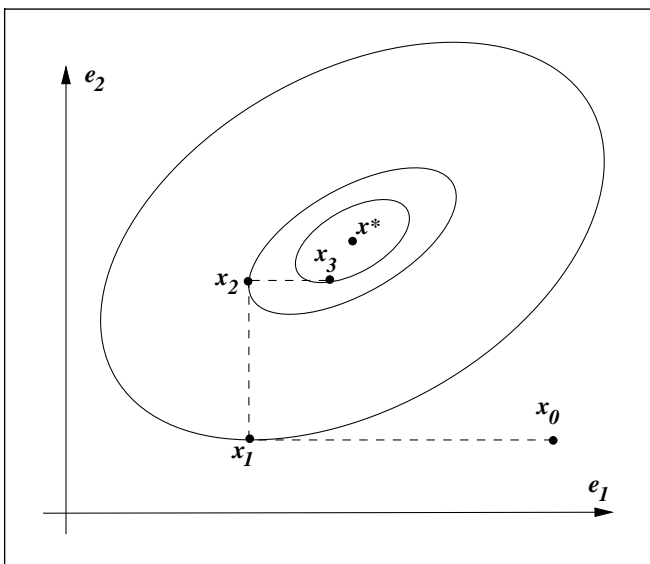


Figure 5.2 Successive minimization along coordinate axes does not find the solution in n iterations, for a general convex quadratic.

e_1, e_2, \dots, e_n in turn. When A is *not* diagonal, its contours are still elliptical, but they are usually no longer aligned with the coordinate directions. The strategy of successive minimization along these directions in turn no longer leads to the solution in n iterations (or even in a finite number of iterations). This phenomenon is illustrated in the two-dimensional example of Figure 5.2. We can, however, recover the nice behavior of Figure 5.1 if we transform the problem to make A diagonal and then minimize along the coordinate directions. Suppose we transform the problem by defining new variables \hat{x} as

$$\hat{x} = S^{-1}x, \quad (5.9)$$

where S is the $n \times n$ matrix defined by

$$S = [p_0 \ p_1 \ \cdots \ p_{n-1}],$$

where $\{p_0, p_1, \dots, p_{n-1}\}$ is the set of conjugate directions with respect to A . The quadratic ϕ defined by (5.2) now becomes

$$\hat{\phi}(\hat{x}) \stackrel{\text{def}}{=} \phi(S\hat{x}) = \frac{1}{2}\hat{x}^T (S^T A S)\hat{x} - (S^T b)^T \hat{x}.$$

By the conjugacy property (5.5), the matrix $S^T A S$ is diagonal, so we can find the minimizing value of $\hat{\phi}$ by performing n one-dimensional minimizations along the coordinate directions

of \hat{x} . Because of the relation (5.9), however, the i th coordinate direction in \hat{x} -space corresponds to the direction p_i in x -space. Hence, the coordinate search strategy applied to $\hat{\phi}$ is equivalent to the conjugate direction algorithm (5.6), (5.7). We conclude, as in Theorem 5.1, that the conjugate direction algorithm terminates in at most n steps.

Returning to Figure 5.1, we note another interesting property: When the Hessian matrix is diagonal, each coordinate minimization correctly determines one of the components of the solution x^* . In other words, after k one-dimensional minimizations, the quadratic has been minimized on the subspace spanned by e_1, e_2, \dots, e_k . The following theorem proves this important result for the general case in which the Hessian of the quadratic is not necessarily diagonal. (Here and later, we use the notation $\text{span}\{p_0, p_1, \dots, p_k\}$ to denote the set of all linear combinations of the vectors p_0, p_1, \dots, p_k .) In proving the result we will make use of the following expression, which is easily verified from the relations (5.4) and (5.6):

$$r_{k+1} = r_k + \alpha_k A p_k. \quad (5.10)$$

Theorem 5.2 (Expanding Subspace Minimization).

Let $x_0 \in \mathbb{R}^n$ be any starting point and suppose that the sequence $\{x_k\}$ is generated by the conjugate direction algorithm (5.6), (5.7). Then

$$r_k^T p_i = 0, \quad \text{for } i = 0, 1, \dots, k-1, \quad (5.11)$$

and x_k is the minimizer of $\phi(x) = \frac{1}{2}x^T A x - b^T x$ over the set

$$\{x \mid x = x_0 + \text{span}\{p_0, p_1, \dots, p_{k-1}\}\}. \quad (5.12)$$

PROOF. We begin by showing that a point \tilde{x} minimizes ϕ over the set (5.12) if and only if $r(\tilde{x})^T p_i = 0$, for each $i = 0, 1, \dots, k-1$. Let us define $h(\sigma) = \phi(x_0 + \sigma_0 p_0 + \dots + \sigma_{k-1} p_{k-1})$, where $\sigma = (\sigma_0, \sigma_1, \dots, \sigma_{k-1})^T$. Since $h(\sigma)$ is a strictly convex quadratic, it has a unique minimizer σ^* that satisfies

$$\frac{\partial h(\sigma^*)}{\partial \sigma_i} = 0, \quad i = 0, 1, \dots, k-1.$$

By the chain rule, this equation implies that

$$\nabla \phi(x_0 + \sigma_0^* p_0 + \dots + \sigma_{k-1}^* p_{k-1})^T p_i = 0, \quad i = 0, 1, \dots, k-1.$$

By recalling the definition (5.3), we have for the minimizer $\tilde{x} = x_0 + \sigma_0^* p_0 + \sigma_1^* p_1 + \dots + \sigma_{k-1}^* p_{k-1}$ on the set (5.12) that $r(\tilde{x})^T p_i = 0$, as claimed.

We now use induction to show that x_k satisfies (5.11). For the case $k = 1$, we have from the fact that $x_1 = x_0 + \alpha_0 p_0$ minimizes ϕ along p_0 that $r_1^T p_0 = 0$. Let us now make

the induction hypothesis, namely, that $r_{k-1}^T p_i = 0$ for $i = 0, 1, \dots, k-2$. By (5.10), we have

$$r_k = r_{k-1} + \alpha_{k-1} A p_{k-1},$$

so that

$$p_{k-1}^T r_k = p_{k-1}^T r_{k-1} + \alpha_{k-1} p_{k-1}^T A p_{k-1} = 0,$$

by the definition (5.7) of α_{k-1} . Meanwhile, for the other vectors p_i , $i = 0, 1, \dots, k-2$, we have

$$p_i^T r_k = p_i^T r_{k-1} + \alpha_{k-1} p_i^T A p_{k-1} = 0,$$

where $p_i^T r_{k-1} = 0$ because of the induction hypothesis and $p_i^T A p_{k-1} = 0$ because of conjugacy of the vectors p_i . We have shown that $r_k^T p_i = 0$, for $i = 0, 1, \dots, k-1$, so the proof is complete. \square

The fact that the current residual r_k is orthogonal to all previous search directions, as expressed in (5.11), is a property that will be used extensively in this chapter.

The discussion so far has been general, in that it applies to a conjugate direction method (5.6), (5.7) based on *any* choice of the conjugate direction set $\{p_0, p_1, \dots, p_{n-1}\}$. There are many ways to choose the set of conjugate directions. For instance, the eigenvectors v_1, v_2, \dots, v_n of A are mutually orthogonal as well as conjugate with respect to A , so these could be used as the vectors $\{p_0, p_1, \dots, p_{n-1}\}$. For large-scale applications, however, computation of the complete set of eigenvectors requires an excessive amount of computation. An alternative approach is to modify the Gram–Schmidt orthogonalization process to produce a set of conjugate directions rather than a set of orthogonal directions. (This modification is easy to produce, since the properties of conjugacy and orthogonality are closely related in spirit.) However, the Gram–Schmidt approach is also expensive, since it requires us to store the entire direction set.

BASIC PROPERTIES OF THE CONJUGATE GRADIENT METHOD

The conjugate gradient method is a conjugate direction method with a very special property: In generating its set of conjugate vectors, it can compute a new vector p_k by using only the previous vector p_{k-1} . It does *not* need to know all the previous elements p_0, p_1, \dots, p_{k-2} of the conjugate set; p_k is automatically conjugate to these vectors. This remarkable property implies that the method requires little storage and computation.

In the conjugate gradient method, each direction p_k is chosen to be a linear combination of the negative residual $-r_k$ (which, by (5.3), is the steepest descent direction for the

function ϕ) and the previous direction p_{k-1} . We write

$$p_k = -r_k + \beta_k p_{k-1}, \quad (5.13)$$

where the scalar β_k is to be determined by the requirement that p_{k-1} and p_k must be conjugate with respect to A . By premultiplying (5.13) by $p_{k-1}^T A$ and imposing the condition $p_{k-1}^T A p_k = 0$, we find that

$$\beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}.$$

We choose the first search direction p_0 to be the steepest descent direction at the initial point x_0 . As in the general conjugate direction method, we perform successive one-dimensional minimizations along each of the search directions. We have thus specified a complete algorithm, which we express formally as follows:

Algorithm 5.1 (CG–Preliminary Version).

Given x_0 ;

Set $r_0 \leftarrow Ax_0 - b$, $p_0 \leftarrow -r_0$, $k \leftarrow 0$;

while $r_k \neq 0$

$$\alpha_k \leftarrow -\frac{r_k^T p_k}{p_k^T A p_k}; \quad (5.14a)$$

$$x_{k+1} \leftarrow x_k + \alpha_k p_k; \quad (5.14b)$$

$$r_{k+1} \leftarrow Ax_{k+1} - b; \quad (5.14c)$$

$$\beta_{k+1} \leftarrow \frac{r_{k+1}^T A p_k}{p_k^T A p_k}; \quad (5.14d)$$

$$p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k; \quad (5.14e)$$

$$k \leftarrow k + 1; \quad (5.14f)$$

end (while)

This version is useful for studying the essential properties of the conjugate gradient method, but we present a more efficient version later. We show first that the directions p_0, p_1, \dots, p_{n-1} are indeed conjugate, which by Theorem 5.1 implies termination in n steps. The theorem below establishes this property and two other important properties. First, the residuals r_i are mutually orthogonal. Second, each search direction p_k and residual r_k is contained in the *Krylov subspace of degree k for r_0* , defined as

$$\mathcal{K}(r_0; k) \stackrel{\text{def}}{=} \text{span}\{r_0, Ar_0, \dots, A^k r_0\}. \quad (5.15)$$

Theorem 5.3.

Suppose that the k th iterate generated by the conjugate gradient method is not the solution point x^* . The following four properties hold:

$$r_k^T r_i = 0, \quad \text{for } i = 0, 1, \dots, k-1, \quad (5.16)$$

$$\text{span}\{r_0, r_1, \dots, r_k\} = \text{span}\{r_0, Ar_0, \dots, A^k r_0\}, \quad (5.17)$$

$$\text{span}\{p_0, p_1, \dots, p_k\} = \text{span}\{r_0, Ar_0, \dots, A^k r_0\}, \quad (5.18)$$

$$p_k^T Ap_i = 0, \quad \text{for } i = 0, 1, \dots, k-1. \quad (5.19)$$

Therefore, the sequence $\{x_k\}$ converges to x^* in at most n steps.

PROOF. The proof is by induction. The expressions (5.17) and (5.18) hold trivially for $k = 0$, while (5.19) holds by construction for $k = 1$. Assuming now that these three expressions are true for some k (the induction hypothesis), we show that they continue to hold for $k + 1$.

To prove (5.17), we show first that the set on the left-hand side is contained in the set on the right-hand side. Because of the induction hypothesis, we have from (5.17) and (5.18) that

$$r_k \in \text{span}\{r_0, Ar_0, \dots, A^k r_0\}, \quad p_k \in \text{span}\{r_0, Ar_0, \dots, A^k r_0\},$$

while by multiplying the second of these expressions by A , we obtain

$$Ap_k \in \text{span}\{Ar_0, \dots, A^{k+1} r_0\}. \quad (5.20)$$

By applying (5.10), we find that

$$r_{k+1} \in \text{span}\{r_0, Ar_0, \dots, A^{k+1} r_0\}.$$

By combining this expression with the induction hypothesis for (5.17), we conclude that

$$\text{span}\{r_0, r_1, \dots, r_k, r_{k+1}\} \subset \text{span}\{r_0, Ar_0, \dots, A^{k+1} r_0\}.$$

To prove that the reverse inclusion holds as well, we use the induction hypothesis on (5.18) to deduce that

$$A^{k+1} r_0 = A(A^k r_0) \in \text{span}\{Ap_0, Ap_1, \dots, Ap_k\}.$$

Since by (5.10) we have $Ap_i = (r_{i+1} - r_i)/\alpha_i$ for $i = 0, 1, \dots, k$, it follows that

$$A^{k+1} r_0 \in \text{span}\{r_0, r_1, \dots, r_{k+1}\}.$$

By combining this expression with the induction hypothesis for (5.17), we find that

$$\text{span}\{r_0, Ar_0, \dots, A^{k+1}r_0\} \subset \text{span}\{r_0, r_1, \dots, r_k, r_{k+1}\}.$$

Therefore, the relation (5.17) continues to hold when k is replaced by $k + 1$, as claimed.

We show that (5.18) continues to hold when k is replaced by $k + 1$ by the following argument:

$$\begin{aligned} & \text{span}\{p_0, p_1, \dots, p_k, p_{k+1}\} \\ &= \text{span}\{p_0, p_1, \dots, p_k, r_{k+1}\} && \text{by (5.14e)} \\ &= \text{span}\{r_0, Ar_0, \dots, A^k r_0, r_{k+1}\} && \text{by induction hypothesis for (5.18)} \\ &= \text{span}\{r_0, r_1, \dots, r_k, r_{k+1}\} && \text{by (5.17)} \\ &= \text{span}\{r_0, Ar_0, \dots, A^{k+1}r_0\} && \text{by (5.17) for } k + 1. \end{aligned}$$

Next, we prove the conjugacy condition (5.19) with k replaced by $k + 1$. By multiplying (5.14e) by Ap_i , $i = 0, 1, \dots, k$, we obtain

$$p_{k+1}^T Ap_i = -r_{k+1}^T Ap_i + \beta_{k+1} p_k^T Ap_i. \quad (5.21)$$

By the definition (5.14d) of β_k , the right-hand-side of (5.21) vanishes when $i = k$. For $i \leq k - 1$ we need to collect a number of observations. Note first that our induction hypothesis for (5.19) implies that the directions p_0, p_1, \dots, p_k are conjugate, so we can apply Theorem 5.2 to deduce that

$$r_{k+1}^T p_i = 0, \quad \text{for } i = 0, 1, \dots, k. \quad (5.22)$$

Second, by repeatedly applying (5.18), we find that for $i = 0, 1, \dots, k - 1$, the following inclusion holds:

$$\begin{aligned} Ap_i &\in A \text{span}\{r_0, Ar_0, \dots, A^i r_0\} = \text{span}\{Ar_0, A^2 r_0, \dots, A^{i+1} r_0\} \\ &\subset \text{span}\{p_0, p_1, \dots, p_{i+1}\}. \end{aligned} \quad (5.23)$$

By combining (5.22) and (5.23), we deduce that

$$r_{k+1}^T Ap_i = 0, \quad \text{for } i = 0, 1, \dots, k - 1,$$

so the first term in the right-hand-side of (5.21) vanishes for $i = 0, 1, \dots, k - 1$. Because of the induction hypothesis for (5.19), the second term vanishes as well, and we

conclude that $p_{k+1}^T A p_i = 0, i = 0, 1, \dots, k$. Hence, the induction argument holds for (5.19) also.

It follows that the direction set generated by the conjugate gradient method is indeed a conjugate direction set, so Theorem 5.1 tells us that the algorithm terminates in at most n iterations.

Finally, we prove (5.16) by a noninductive argument. Because the direction set is conjugate, we have from (5.11) that $r_k^T p_i = 0$ for all $i = 0, 1, \dots, k - 1$ and any $k = 1, 2, \dots, n - 1$. By rearranging (5.14e), we find that

$$p_i = -r_i + \beta_i p_{i-1},$$

so that $r_i \in \text{span}\{p_i, p_{i-1}\}$ for all $i = 1, \dots, k - 1$. We conclude that $r_k^T r_i = 0$ for all $i = 1, \dots, k - 1$. To complete the proof, we note that $r_k^T r_0 = -r_k^T p_0 = 0$, by definition of p_0 in Algorithm 5.1 and by (5.11). \square

The proof of this theorem relies on the fact that the first direction p_0 is the steepest descent direction $-r_0$; in fact, the result does not hold for other choices of p_0 . Since the gradients r_k are mutually orthogonal, the term “conjugate gradient method” is actually a misnomer. It is the search directions, not the gradients, that are conjugate with respect to A .

A PRACTICAL FORM OF THE CONJUGATE GRADIENT METHOD

We can derive a slightly more economical form of the conjugate gradient method by using the results of Theorems 5.2 and 5.3. First, we can use (5.14e) and (5.11) to replace the formula (5.14a) for α_k by

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}.$$

Second, we have from (5.10) that $\alpha_k A p_k = r_{k+1} - r_k$, so by applying (5.14e) and (5.11) once again we can simplify the formula for β_{k+1} to

$$\beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}.$$

By using these formulae together with (5.10), we obtain the following standard form of the conjugate gradient method.

Algorithm 5.2 (CG).Given x_0 ;Set $r_0 \leftarrow Ax_0 - b$, $p_0 \leftarrow -r_0$, $k \leftarrow 0$;**while** $r_k \neq 0$

$$\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}; \quad (5.24a)$$

$$x_{k+1} \leftarrow x_k + \alpha_k p_k; \quad (5.24b)$$

$$r_{k+1} \leftarrow r_k + \alpha_k A p_k; \quad (5.24c)$$

$$\beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}; \quad (5.24d)$$

$$p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k; \quad (5.24e)$$

$$k \leftarrow k + 1; \quad (5.24f)$$

end (while)

At any given point in Algorithm 5.2 we never need to know the vectors x , r , and p for more than the last two iterations. Accordingly, implementations of this algorithm overwrite old values of these vectors to save on storage. The major computational tasks to be performed at each step are computation of the matrix–vector product $A p_k$, calculation of the inner products $p_k^T (A p_k)$ and $r_{k+1}^T r_{k+1}$, and calculation of three vector sums. The inner product and vector sum operations can be performed in a small multiple of n floating-point operations, while the cost of the matrix–vector product is, of course, dependent on the problem. The CG method is recommended only for large problems; otherwise, Gaussian elimination or other factorization algorithms such as the singular value decomposition are to be preferred, since they are less sensitive to rounding errors. For large problems, the CG method has the advantage that it does not alter the coefficient matrix and (in contrast to factorization techniques) does not produce fill in the arrays holding the matrix. Another key property is that the CG method sometimes approaches the solution quickly, as we discuss next.

RATE OF CONVERGENCE

We have seen that in exact arithmetic the conjugate gradient method will terminate at the solution in at most n iterations. What is more remarkable is that when the distribution of the eigenvalues of A has certain favorable features, the algorithm will identify the solution in many fewer than n iterations. To explain this property, we begin by viewing the expanding subspace minimization property proved in Theorem 5.2 in a slightly different way, using it to show that Algorithm 5.2 is optimal in a certain important sense.

From (5.24b) and (5.18), we have that

$$\begin{aligned} x_{k+1} &= x_0 + \alpha_0 p_0 + \cdots + \alpha_k p_k \\ &= x_0 + \gamma_0 r_0 + \gamma_1 A r_0 + \cdots + \gamma_k A^k r_0, \end{aligned} \quad (5.25)$$

for some constants γ_i . We now define $P_k^*(\cdot)$ to be a polynomial of degree k with coefficients $\gamma_0, \gamma_1, \dots, \gamma_k$. Like any polynomial, P_k^* can take either a scalar or a square matrix as its argument. For the matrix argument A , we have

$$P_k^*(A) = \gamma_0 I + \gamma_1 A + \cdots + \gamma_k A^k,$$

which allows us to express (5.25) as follows:

$$x_{k+1} = x_0 + P_k^*(A)r_0. \quad (5.26)$$

We now show that among all possible methods whose first k steps are restricted to the Krylov subspace $\mathcal{K}(r_0; k)$ given by (5.15), Algorithm 5.2 does the best job of minimizing the distance to the solution after k steps, when this distance is measured by the weighted norm measure $\|\cdot\|_A$ defined by

$$\|z\|_A^2 = z^T A z. \quad (5.27)$$

(Recall that this norm was used in the analysis of the steepest descent method of Chapter 3.) Using this norm and the definition of ϕ (5.2), and the fact that x^* minimizes ϕ , it is easy to show that

$$\frac{1}{2} \|x - x^*\|_A^2 = \frac{1}{2} (x - x^*)^T A (x - x^*) = \phi(x) - \phi(x^*). \quad (5.28)$$

Theorem 5.2 states that x_{k+1} minimizes ϕ , and hence $\|x - x^*\|_A^2$, over the set $x_0 + \text{span}\{p_0, p_1, \dots, p_k\}$, which by (5.18) is the same as $x_0 + \text{span}\{r_0, A r_0, \dots, A^k r_0\}$. It follows from (5.26) that the polynomial P_k^* solves the following problem in which the minimum is taken over the space of all possible polynomials of degree k :

$$\min_{P_k} \|x_0 + P_k(A)r_0 - x^*\|_A. \quad (5.29)$$

We exploit this optimality property repeatedly in the remainder of the section.

Since

$$r_0 = Ax_0 - b = Ax_0 - Ax^* = A(x_0 - x^*),$$

we have that

$$x_{k+1} - x^* = x_0 + P_k^*(A)r_0 - x^* = [I + P_k^*(A)A](x_0 - x^*). \quad (5.30)$$

Let $0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ be the eigenvalues of A , and let v_1, v_2, \dots, v_n be the corresponding orthonormal eigenvectors, so that

$$A = \sum_{i=1}^n \lambda_i v_i v_i^T.$$

Since the eigenvectors span the whole space \mathbb{R}^n , we can write

$$x_0 - x^* = \sum_{i=1}^n \xi_i v_i, \quad (5.31)$$

for some coefficients ξ_i . It is easy to show that any eigenvector of A is also an eigenvector of $P_k(A)$ for any polynomial P_k . For our particular matrix A and its eigenvalues λ_i and eigenvectors v_i , we have

$$P_k(A)v_i = P_k(\lambda_i)v_i, \quad i = 1, 2, \dots, n.$$

By substituting (5.31) into (5.30) we have

$$x_{k+1} - x^* = \sum_{i=1}^n [1 + \lambda_i P_k^*(\lambda_i)] \xi_i v_i.$$

By using the fact that $\|z\|_A^2 = z^T A z = \sum_{i=1}^n \lambda_i (v_i^T z)^2$, we have

$$\|x_{k+1} - x^*\|_A^2 = \sum_{i=1}^n \lambda_i [1 + \lambda_i P_k^*(\lambda_i)]^2 \xi_i^2. \quad (5.32)$$

Since the polynomial P_k^* generated by the CG method is optimal with respect to this norm, we have

$$\|x_{k+1} - x^*\|_A^2 = \min_{P_k} \sum_{i=1}^n \lambda_i [1 + \lambda_i P_k(\lambda_i)]^2 \xi_i^2.$$

By extracting the largest of the terms $[1 + \lambda_i P_k(\lambda_i)]^2$ from this expression, we obtain that

$$\begin{aligned} \|x_{k+1} - x^*\|_A^2 &\leq \min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2 \left(\sum_{j=1}^n \lambda_j \xi_j^2 \right) \\ &= \min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2 \|x_0 - x^*\|_A^2, \end{aligned} \quad (5.33)$$

where we have used the fact that $\|x_0 - x^*\|_A^2 = \sum_{j=1}^n \lambda_j \xi_j^2$.

The expression (5.33) allows us to quantify the convergence rate of the CG method by estimating the nonnegative scalar quantity

$$\min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2. \quad (5.34)$$

In other words, we search for a polynomial P_k that makes this expression as small as possible. In some practical cases, we can find this polynomial explicitly and draw some interesting conclusions about the properties of the CG method. The following result is an example.

Theorem 5.4.

If A has only r distinct eigenvalues, then the CG iteration will terminate at the solution in at most r iterations.

PROOF. Suppose that the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ take on the r distinct values $\tau_1 < \tau_2 < \dots < \tau_r$. We define a polynomial $Q_r(\lambda)$ by

$$Q_r(\lambda) = \frac{(-1)^r}{\tau_1 \tau_2 \cdots \tau_r} (\lambda - \tau_1)(\lambda - \tau_2) \cdots (\lambda - \tau_r),$$

and note that $Q_r(\lambda_i) = 0$ for $i = 1, 2, \dots, n$ and $Q_r(0) = 1$. From the latter observation, we deduce that $Q_r(\lambda) - 1$ is a polynomial of degree r with a root at $\lambda = 0$, so by polynomial division, the function \bar{P}_{r-1} defined by

$$\bar{P}_{r-1}(\lambda) = (Q_r(\lambda) - 1)/\lambda$$

is a polynomial of degree $r - 1$. By setting $k = r - 1$ in (5.34), we have

$$0 \leq \min_{P_{r-1}} \max_{1 \leq i \leq n} [1 + \lambda_i P_{r-1}(\lambda_i)]^2 \leq \max_{1 \leq i \leq n} [1 + \lambda_i \bar{P}_{r-1}(\lambda_i)]^2 = \max_{1 \leq i \leq n} Q_r^2(\lambda_i) = 0.$$

Hence, the constant in (5.34) is zero for the value $k = r - 1$, so we have by substituting into (5.33) that $\|x_r - x^*\|_A^2 = 0$, and therefore $x_r = x^*$, as claimed. \square

By using similar reasoning, Luenberger [195] establishes the following estimate, which gives a useful characterization of the behavior of the CG method.

Theorem 5.5.

If A has eigenvalues $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$, we have that

$$\|x_{k+1} - x^*\|_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1} \right)^2 \|x_0 - x^*\|_A^2. \quad (5.35)$$

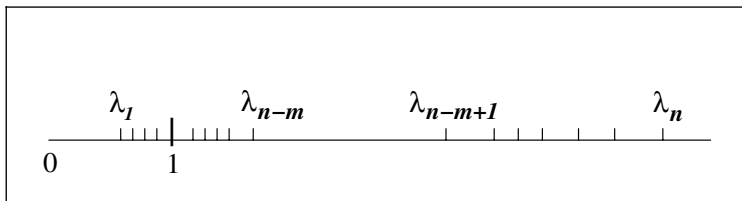


Figure 5.3 Two clusters of eigenvalues.

Without giving details of the proof, we describe how this result is obtained from (5.33). One selects a polynomial \bar{P}_k of degree k such that the polynomial $Q_{k+1}(\lambda) = 1 + \lambda \bar{P}_k(\lambda)$ has roots at the k largest eigenvalues $\lambda_n, \lambda_{n-1}, \dots, \lambda_{n-k+1}$, as well as at the midpoint between λ_1 and λ_{n-k} . It can be shown that the maximum value attained by Q_{k+1} on the remaining eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_{n-k}$ is precisely $(\lambda_{n-k} - \lambda_1)/(\lambda_{n-k} + \lambda_1)$.

We now illustrate how Theorem 5.5 can be used to predict the behavior of the CG method on specific problems. Suppose we have the situation plotted in Figure 5.3, where the eigenvalues of A consist of m large values, with the remaining $n - m$ smaller eigenvalues clustered around 1. If we define $\epsilon = \lambda_{n-m} - \lambda_1$, Theorem 5.5 tells us that after $m + 1$ steps of the conjugate gradient algorithm, we have

$$\|x_{m+1} - x^*\|_A \approx \epsilon \|x_0 - x^*\|_A.$$

For a small value of ϵ , we conclude that the CG iterates will provide a good estimate of the solution after only $m + 1$ steps.

Figure 5.4 shows the behavior of CG on a problem of this type, which has five large eigenvalues with all the smaller eigenvalues clustered between 0.95 and 1.05, and compares this behavior with that of CG on a problem in which the eigenvalues satisfy some random distribution. In both cases, we plot the log of ϕ after each iteration.

For the problem with clustered eigenvalues, Theorem 5.5 predicts a sharp decrease in the error measure at iteration 6. Note, however, that this decrease was achieved one iteration earlier, illustrating the fact that Theorem 5.5 gives only an upper bound, and that the rate of convergence can be faster. By contrast, we observe in Figure 5.4 that for the problem with randomly distributed eigenvalues (dashed line), the convergence rate is slower and more uniform.

Figure 5.4 illustrates another interesting feature: After one more iteration (a total of seven) on the problem with clustered eigenvalues, the error measure drops sharply. An extension of the arguments leading to Theorem 5.4 explains this behavior. It is *almost* true to say that the matrix A has just six distinct eigenvalues: the five large eigenvalues and 1. Then we would expect the error measure to be zero after six iterations. Because the eigenvalues near 1 are slightly spread out, however, the error does not become very small until iteration 7.

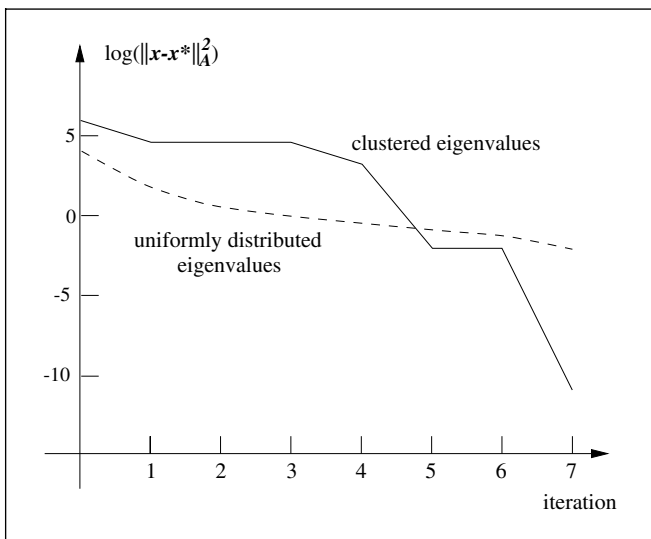


Figure 5.4 Performance of the conjugate gradient method on (a) a problem in which five of the eigenvalues are large and the remainder are clustered near 1, and (b) a matrix with uniformly distributed eigenvalues.

To state this claim more precisely, it is generally true that if the eigenvalues occur in r distinct clusters, the CG iterates will *approximately* solve the problem in about r steps (see [136]). This result can be proved by constructing a polynomial \tilde{P}_{r-1} such that $(1 + \lambda \tilde{P}_{r-1}(\lambda))$ has zeros inside each of the clusters. This polynomial may not vanish at the eigenvalues λ_i , $i = 1, 2, \dots, n$, but its value will be small at these points, so the constant defined in (5.34) will be small for $k \geq r - 1$. We illustrate this behavior in Figure 5.5, which shows the performance of CG on a matrix of dimension $n = 14$ that has four clusters of eigenvalues: single eigenvalues at 140 and 120, a cluster of 10 eigenvalues very close to 10, with the remaining eigenvalues clustered between 0.95 and 1.05. After four iterations, the error has decreased significantly. After six iterations, the solution is identified to good accuracy.

Another, more approximate, convergence expression for CG is based on the Euclidean condition number of A , which is defined by

$$\kappa(A) = \|A\|_2 \|A^{-1}\|_2 = \lambda_n / \lambda_1.$$

It can be shown that

$$\|x_k - x^*\|_A \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \|x_0 - x^*\|_A. \quad (5.36)$$

This bound often gives a large overestimate of the error, but it can be useful in those cases

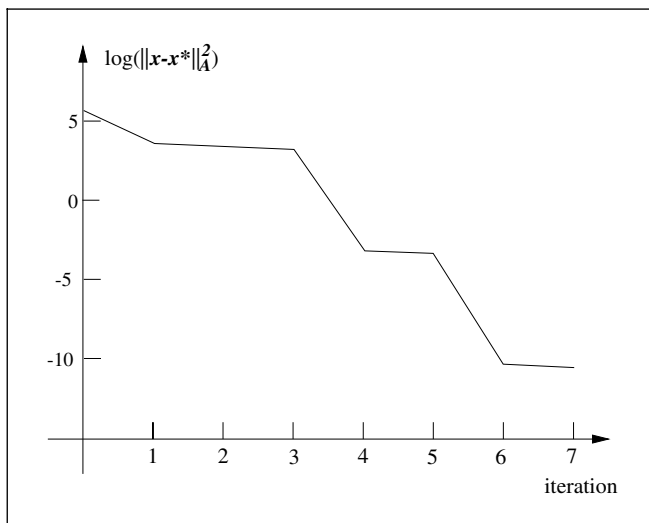


Figure 5.5 Performance of the conjugate gradient method on a matrix in which the eigenvalues occur in four distinct clusters.

where the only information we have about A is estimates of the extreme eigenvalues λ_1 and λ_n . This bound should be compared with that of the steepest descent method given by (3.29), which is identical in form but which depends on the condition number $\kappa(A)$, and not on its square root $\sqrt{\kappa(A)}$.

PRECONDITIONING

We can accelerate the conjugate gradient method by transforming the linear system to improve the eigenvalue distribution of A . The key to this process, which is known as *preconditioning*, is a change of variables from x to \hat{x} via a nonsingular matrix C , that is,

$$\hat{x} = Cx. \quad (5.37)$$

The quadratic ϕ defined by (5.2) is transformed accordingly to

$$\hat{\phi}(\hat{x}) = \frac{1}{2}\hat{x}^T(C^{-T}AC^{-1})\hat{x} - (C^{-T}b)^T\hat{x}. \quad (5.38)$$

If we use Algorithm 5.2 to minimize $\hat{\phi}$ or, equivalently, to solve the linear system

$$(C^{-T}AC^{-1})\hat{x} = C^{-T}b,$$

then the convergence rate will depend on the eigenvalues of the matrix $C^{-T}AC^{-1}$ rather than those of A . Therefore, we aim to choose C such that the eigenvalues of $C^{-T}AC^{-1}$

are more favorable for the convergence theory discussed above. We can try to choose C such that the condition number of $C^{-T}AC^{-1}$ is much smaller than the original condition number of A , for instance, so that the constant in (5.36) is smaller. We could also try to choose C such that the eigenvalues of $C^{-T}AC^{-1}$ are clustered, which by the discussion of the previous section ensures that the number of iterates needed to find a good approximate solution is not much larger than the number of clusters.

It is not necessary to carry out the transformation (5.37) explicitly. Rather, we can apply Algorithm 5.2 to the problem (5.38), in terms of the variables \hat{x} , and then invert the transformations to reexpress all the equations in terms of x . This process of derivation results in Algorithm 5.3 (Preconditioned Conjugate Gradient), which we now define. It happens that Algorithm 5.3 does not make use of C explicitly, but rather the matrix $M = C^T C$, which is symmetric and positive definite by construction.

Algorithm 5.3 (Preconditioned CG).

Given x_0 , preconditioner M ;
 Set $r_0 \leftarrow Ax_0 - b$;
 Solve $My_0 = r_0$ for y_0 ;
 Set $p_0 = -y_0, k \leftarrow 0$;
while $r_k \neq 0$

$$\alpha_k \leftarrow \frac{r_k^T y_k}{p_k^T A p_k}; \quad (5.39a)$$

$$x_{k+1} \leftarrow x_k + \alpha_k p_k; \quad (5.39b)$$

$$r_{k+1} \leftarrow r_k + \alpha_k A p_k; \quad (5.39c)$$

$$\text{Solve } M y_{k+1} = r_{k+1}; \quad (5.39d)$$

$$\beta_{k+1} \leftarrow \frac{r_{k+1}^T y_{k+1}}{r_k^T y_k}; \quad (5.39e)$$

$$p_{k+1} \leftarrow -y_{k+1} + \beta_{k+1} p_k; \quad (5.39f)$$

$$k \leftarrow k + 1; \quad (5.39g)$$

end (while)

If we set $M = I$ in Algorithm 5.3, we recover the standard CG method, Algorithm 5.2. The properties of Algorithm 5.2 generalize to this case in interesting ways. In particular, the orthogonality property (5.16) of the successive residuals becomes

$$r_i^T M^{-1} r_j = 0 \quad \text{for all } i \neq j. \quad (5.40)$$

In terms of computational effort, the main difference between the preconditioned and unpreconditioned CG methods is the need to solve systems of the form $My = r$ (step (5.39d)).

PRACTICAL PRECONDITIONERS

No single preconditioning strategy is “best” for all conceivable types of matrices: The tradeoff between various objectives—effectiveness of M , inexpensive computation and storage of M , inexpensive solution of $My = r$ —varies from problem to problem.

Good preconditioning strategies have been devised for specific types of matrices, in particular, those arising from discretizations of partial differential equations (PDEs). Often, the preconditioner is defined in such a way that the system $My = r$ amounts to a simplified version of the original system $Ax = b$. In the case of a PDE, $My = r$ could represent a coarser discretization of the underlying continuous problem than $Ax = b$. As in many other areas of optimization and numerical analysis, knowledge about the structure and origin of a problem (in this case, knowledge that the system $Ax = b$ is a finite-dimensional representation of a PDE) is the key to devising effective techniques for solving the problem.

General-purpose preconditioners have also been proposed, but their success varies greatly from problem to problem. The most important strategies of this type include symmetric successive overrelaxation (SSOR), incomplete Cholesky, and banded preconditioners. (See [272], [136], and [72] for discussions of these techniques.) *Incomplete Cholesky* is probably the most effective in general. The basic idea is simple: We follow the Cholesky procedure, but instead of computing the exact Cholesky factor L that satisfies $A = LL^T$, we compute an approximate factor \tilde{L} that is sparser than L . (Usually, we require \tilde{L} to be no denser, or not much denser, than the lower triangle of the original matrix A .) We then have $A \approx \tilde{L}\tilde{L}^T$, and by choosing $C = \tilde{L}^T$, we obtain $M = \tilde{L}\tilde{L}^T$ and

$$C^{-T}AC^{-1} = \tilde{L}^{-1}A\tilde{L}^{-T} \approx I,$$

so the eigenvalue distribution of $C^{-T}AC^{-1}$ is favorable. We do not compute M explicitly, but rather store the factor \tilde{L} and solve the system $My = r$ by performing two triangular substitutions with \tilde{L} . Because the sparsity of \tilde{L} is similar to that of A , the cost of solving $My = r$ is similar to the cost of computing the matrix–vector product Ap .

There are several possible pitfalls in the incomplete Cholesky approach. One is that the resulting matrix may not be (sufficiently) positive definite, and in this case one may need to increase the values of the diagonal elements to ensure that a value for \tilde{L} can be found. Numerical instability or breakdown can occur during the incomplete factorization because of the sparsity conditions we impose on the factor \tilde{L} . This difficulty can be remedied by allowing additional fill-in in \tilde{L} , but the denser factor will be more expensive to compute and to apply at each iteration.

5.2 NONLINEAR CONJUGATE GRADIENT METHODS

We have noted that the CG method, Algorithm 5.2, can be viewed as a minimization algorithm for the convex quadratic function ϕ defined by (5.2). It is natural to ask whether we can adapt the approach to minimize general convex functions, or even general nonlinear functions f . In fact, as we show in this section, nonlinear variants of the conjugate gradient are well studied and have proved to be quite successful in practice.

THE FLETCHER–REEVES METHOD

Fletcher and Reeves [107] showed how to extend the conjugate gradient method to nonlinear functions by making two simple changes in Algorithm 5.2. First, in place of the formula (5.24a) for the step length α_k (which minimizes ϕ along the search direction p_k), we need to perform a line search that identifies an approximate minimum of the nonlinear function f along p_k . Second, the residual r , which is simply the gradient of ϕ in Algorithm 5.2 (see (5.3)), must be replaced by the gradient of the nonlinear objective f . These changes give rise to the following algorithm for nonlinear optimization.

Algorithm 5.4 (FR).

Given x_0 ;

Evaluate $f_0 = f(x_0)$, $\nabla f_0 = \nabla f(x_0)$;

Set $p_0 \leftarrow -\nabla f_0$, $k \leftarrow 0$;

while $\nabla f_k \neq 0$

Compute α_k and set $x_{k+1} = x_k + \alpha_k p_k$;

Evaluate ∇f_{k+1} ;

$$\beta_{k+1}^{\text{FR}} \leftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}; \quad (5.41a)$$

$$p_{k+1} \leftarrow -\nabla f_{k+1} + \beta_{k+1}^{\text{FR}} p_k; \quad (5.41b)$$

$$k \leftarrow k + 1; \quad (5.41c)$$

end (while)

If we choose f to be a strongly convex quadratic and α_k to be the exact minimizer, this algorithm reduces to the linear conjugate gradient method, Algorithm 5.2. Algorithm 5.4 is appealing for large nonlinear optimization problems because each iteration requires only evaluation of the objective function and its gradient. No matrix operations are required for the step computation, and just a few vectors of storage are required.

To make the specification of Algorithm 5.4 complete, we need to be more precise about the choice of line search parameter α_k . Because of the second term in (5.41b), the search direction p_k may fail to be a descent direction unless α_k satisfies certain conditions.

By taking the inner product of (5.41b) (with k replacing $k + 1$) with the gradient vector ∇f_k , we obtain

$$\nabla f_k^T p_k = -\|\nabla f_k\|^2 + \beta_k^{\text{FR}} \nabla f_k^T p_{k-1}. \quad (5.42)$$

If the line search is exact, so that α_{k-1} is a local minimizer of f along the direction p_{k-1} , we have that $\nabla f_k^T p_{k-1} = 0$. In this case we have from (5.42) that $\nabla f_k^T p_k < 0$, so that p_k is indeed a descent direction. If the line search is not exact, however, the second term in (5.42) may dominate the first term, and we may have $\nabla f_k^T p_k > 0$, implying that p_k is actually a direction of ascent. Fortunately, we can avoid this situation by requiring the step length α_k to satisfy the *strong* Wolfe conditions, which we restate here:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, \quad (5.43a)$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq -c_2 \nabla f_k^T p_k, \quad (5.43b)$$

where $0 < c_1 < c_2 < \frac{1}{2}$. (Note that we impose $c_2 < \frac{1}{2}$ here, in place of the looser condition $c_2 < 1$ that was used in the earlier statement (3.7).) By applying Lemma 5.6 below, we can show that condition (5.43b) implies that (5.42) is negative, and we conclude that any line search procedure that yields an α_k satisfying (5.43) will ensure that all directions p_k are descent directions for the function f .

THE POLAK–RIBIÈRE METHOD AND VARIANTS

There are many variants of the Fletcher–Reeves method that differ from each other mainly in the choice of the parameter β_k . An important variant, proposed by Polak and Ribière, defines this parameter as follows:

$$\beta_{k+1}^{\text{PR}} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}. \quad (5.44)$$

We refer to the algorithm in which (5.44) replaces (5.41a) as Algorithm PR. It is identical to Algorithm FR when f is a strongly convex quadratic function and the line search is exact, since by (5.16) the gradients are mutually orthogonal, and so $\beta_{k+1}^{\text{PR}} = \beta_{k+1}^{\text{FR}}$. When applied to general nonlinear functions with inexact line searches, however, the behavior of the two algorithms differs markedly. Numerical experience indicates that Algorithm PR tends to be the more robust and efficient of the two.

A surprising fact about Algorithm PR is that the strong Wolfe conditions (5.43) do not guarantee that p_k is always a descent direction. If we define the β parameter as

$$\beta_{k+1}^+ = \max\{\beta_{k+1}^{\text{PR}}, 0\}, \quad (5.45)$$

giving rise to an algorithm we call Algorithm PR+, then a simple adaptation of the strong Wolfe conditions ensures that the descent property holds.

There are many other choices for β_{k+1} that coincide with the Fletcher–Reeves formula β_{k+1}^{FR} in the case where the objective is quadratic and the line search is exact. The Hestenes–Stiefel formula, which defines

$$\beta_{k+1}^{\text{HS}} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{(\nabla f_{k+1} - \nabla f_k)^T p_k}, \quad (5.46)$$

gives rise to an algorithm (called Algorithm HS) that is similar to Algorithm PR, both in terms of its theoretical convergence properties and in its practical performance. Formula (5.46) can be derived by demanding that consecutive search directions be conjugate with respect to the *average Hessian* over the line segment $[x_k, x_{k+1}]$, which is defined as

$$\bar{G}_k \equiv \int_0^1 [\nabla^2 f(x_k + \tau \alpha_k p_k)] d\tau.$$

Recalling from Taylor’s theorem (Theorem 2.1) that $\nabla f_{k+1} = \nabla f_k + \alpha_k \bar{G}_k p_k$, we see that for any direction of the form $p_{k+1} = -\nabla f_{k+1} + \beta_{k+1} p_k$, the condition $p_{k+1}^T \bar{G}_k p_k = 0$ requires β_{k+1} to be given by (5.46).

Later, we see that it is possible to guarantee global convergence for any parameter β_k satisfying the bound

$$|\beta_k| \leq \beta_k^{\text{FR}}, \quad (5.47)$$

for all $k \geq 2$. This fact suggests the following modification of the PR method, which has performed well on some applications. For all $k \geq 2$ let

$$\beta_k = \begin{cases} -\beta_k^{\text{FR}} & \text{if } \beta_k^{\text{PR}} < -\beta_k^{\text{FR}} \\ \beta_k^{\text{PR}} & \text{if } |\beta_k^{\text{PR}}| \leq \beta_k^{\text{FR}} \\ \beta_k^{\text{FR}} & \text{if } \beta_k^{\text{PR}} > \beta_k^{\text{FR}}. \end{cases} \quad (5.48)$$

The algorithm based on this strategy will be denoted by FR-PR.

Other variants of the CG method have recently been proposed. Two choices for β_{k+1} that possess attractive theoretical and computational properties are

$$\beta_{k+1} = \frac{\|\nabla f_{k+1}\|^2}{(\nabla f_{k+1} - \nabla f_k)^T p_k} \quad (5.49)$$

(see [85]) and

$$\beta_{k+1} = \left(\hat{y}_k - 2p_k \frac{\|\hat{y}_k\|^2}{\hat{y}_k^T p_k} \right)^T \frac{\nabla f_{k+1}}{\hat{y}_k^T p_k}, \quad \text{with} \quad \hat{y}_k = \nabla f_{k+1} - \nabla f_k \quad (5.50)$$

(see [161]). These two choices guarantee that p_k is a descent direction, provided the steplength α_k satisfies the Wolfe conditions. The CG algorithms based on (5.49) or (5.50) appear to be competitive with the Polak–Ribière method.

QUADRATIC TERMINATION AND RESTARTS

Implementations of nonlinear conjugate gradient methods usually preserve their close connections with the linear conjugate gradient method. Usually, a quadratic (or cubic) interpolation along the search direction p_k is incorporated into the line search procedure; see Chapter 3. This feature guarantees that when f is a strictly convex quadratic, the step length α_k is chosen to be the exact one-dimensional minimizer, so that the nonlinear conjugate gradient method reduces to the linear method, Algorithm 5.2.

Another modification that is often used in nonlinear conjugate gradient procedures is to *restart* the iteration at every n steps by setting $\beta_k = 0$ in (5.41a), that is, by taking a steepest descent step. Restarting serves to periodically refresh the algorithm, erasing old information that may not be beneficial. We can even prove a strong theoretical result about restarting: It leads to n -step quadratic convergence, that is,

$$\|x_{k+n} - x\| = O(\|x_k - x^*\|^2). \quad (5.51)$$

After a little thought, this result is not so surprising. Consider a function f that is strongly convex quadratic in a neighborhood of the solution, but is nonquadratic everywhere else. Assuming that the algorithm is converging to the solution in question, the iterates will eventually enter the quadratic region. At some point, the algorithm will be restarted in that region, and from that point onward, its behavior will simply be that of the linear conjugate gradient method, Algorithm 5.2. In particular, finite termination will occur within n steps of the restart. The restart is important, because the finite-termination property and other appealing properties of Algorithm 5.2 hold only when its initial search direction p_0 is equal to the negative gradient.

Even if the function f is not exactly quadratic in the region of a solution, Taylor's theorem (Theorem 2.1) implies that it can still be approximated quite closely by a quadratic, provided that it is smooth. Therefore, while we would not expect termination in n steps after the restart, it is not surprising that substantial progress is made toward the solution, as indicated by the expression (5.51).

Though the result (5.51) is interesting from a theoretical viewpoint, it may not be relevant in a practical context, because nonlinear conjugate gradient methods can be recommended only for solving problems with large n . Restarts may never occur in such problems because an approximate solution may be located in fewer than n steps. Hence, nonlinear CG method are sometimes implemented without restarts, or else they include strategies for restarting that are based on considerations other than iteration counts. The most popular restart strategy makes use of the observation (5.16), which is that the gradients are mutually orthogonal when f is a quadratic function. A restart is performed whenever two consecutive

gradients are far from orthogonal, as measured by the test

$$\frac{|\nabla f_k^T \nabla f_{k-1}|}{\|\nabla f_k\|^2} \geq \nu, \quad (5.52)$$

where a typical value for the parameter ν is 0.1.

We could also think of formula (5.45) as a restarting strategy, because p_{k+1} will revert to the steepest descent direction whenever β_k^{PR} is negative. In contrast to (5.52), these restarts are rather infrequent because β_k^{PR} is positive most of the time.

BEHAVIOR OF THE FLETCHER–REEVES METHOD

We now investigate the Fletcher–Reeves algorithm, Algorithm 5.4, a little more closely, proving that it is globally convergent and explaining some of its observed inefficiencies.

The following result gives conditions on the line search under which all search directions are descent directions. It assumes that the level set $\mathcal{L} = \{x : f(x) \leq f(x_0)\}$ is bounded and that f is twice continuously differentiable, so that we have from Lemma 3.1 that there exists a step length α_k satisfying the strong Wolfe conditions.

Lemma 5.6.

Suppose that Algorithm 5.4 is implemented with a step length α_k that satisfies the strong Wolfe conditions (5.43) with $0 < c_2 < \frac{1}{2}$. Then the method generates descent directions p_k that satisfy the following inequalities:

$$-\frac{1}{1-c_2} \leq \frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2} \leq \frac{2c_2-1}{1-c_2}, \quad \text{for all } k = 0, 1, \dots \quad (5.53)$$

PROOF. Note first that the function $t(\xi) \stackrel{\text{def}}{=} (2\xi - 1)/(1 - \xi)$ is monotonically increasing on the interval $[0, \frac{1}{2}]$ and that $t(0) = -1$ and $t(\frac{1}{2}) = 0$. Hence, because of $c_2 \in (0, \frac{1}{2})$, we have

$$-1 < \frac{2c_2-1}{1-c_2} < 0. \quad (5.54)$$

The descent condition $\nabla f_k^T p_k < 0$ follows immediately once we establish (5.53).

The proof is by induction. For $k = 0$, the middle term in (5.53) is -1 , so by using (5.54), we see that both inequalities in (5.53) are satisfied. Next, assume that (5.53) holds for some $k \geq 1$. From (5.41b) and (5.41a) we have

$$\frac{\nabla f_{k+1}^T p_{k+1}}{\|\nabla f_{k+1}\|^2} = -1 + \beta_{k+1} \frac{\nabla f_{k+1}^T p_k}{\|\nabla f_{k+1}\|^2} = -1 + \frac{\nabla f_{k+1}^T p_k}{\|\nabla f_k\|^2}. \quad (5.55)$$

By using the line search condition (5.43b), we have

$$|\nabla f_{k+1}^T p_k| \leq -c_2 \nabla f_k^T p_k,$$

so by combining with (5.55) and recalling (5.41a), we obtain

$$-1 + c_2 \frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2} \leq \frac{\nabla f_{k+1}^T p_{k+1}}{\|\nabla f_{k+1}\|^2} \leq -1 - c_2 \frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2}.$$

Substituting for the term $\nabla f_k^T p_k / \|\nabla f_k\|^2$ from the left-hand-side of the induction hypothesis (5.53), we obtain

$$-1 - \frac{c_2}{1 - c_2} \leq \frac{\nabla f_{k+1}^T p_{k+1}}{\|\nabla f_{k+1}\|^2} \leq -1 + \frac{c_2}{1 - c_2},$$

which shows that (5.53) holds for $k + 1$ as well. \square

This result used only the second strong Wolfe condition (5.43b); the first Wolfe condition (5.43a) will be needed in the next section to establish global convergence. The bounds on $\nabla f_k^T p_k$ in (5.53) impose a limit on how fast the norms of the steps $\|p_k\|$ can grow, and they will play a crucial role in the convergence analysis given below.

Lemma 5.6 can also be used to explain a weakness of the Fletcher–Reeves method. We will argue that if the method generates a bad direction and a tiny step, then the next direction and next step are also likely to be poor. As in Chapter 3, we let θ_k denote the angle between p_k and the steepest descent direction $-\nabla f_k$, defined by

$$\cos \theta_k = \frac{-\nabla f_k^T p_k}{\|\nabla f_k\| \|p_k\|}. \quad (5.56)$$

Suppose that p_k is a poor search direction, in the sense that it makes an angle of nearly 90° with $-\nabla f_k$, that is, $\cos \theta_k \approx 0$. By multiplying both sides of (5.53) by $\|\nabla f_k\| / \|p_k\|$ and using (5.56), we obtain

$$\frac{1 - 2c_2 \|\nabla f_k\|}{1 - c_2 \|p_k\|} \leq \cos \theta_k \leq \frac{1}{1 - c_2} \frac{\|\nabla f_k\|}{\|p_k\|}, \quad \text{for all } k = 0, 1, \dots \quad (5.57)$$

From these inequalities, we deduce that $\cos \theta_k \approx 0$ if and only if

$$\|\nabla f_k\| \ll \|p_k\|.$$

Since p_k is almost orthogonal to the gradient, it is likely that the step from x_k to x_{k+1} is tiny, that is, $x_{k+1} \approx x_k$. If so, we have $\nabla f_{k+1} \approx \nabla f_k$, and therefore

$$\beta_{k+1}^{\text{FR}} \approx 1, \quad (5.58)$$

by the definition (5.41a). By using this approximation together with $\|\nabla f_{k+1}\| \approx \|\nabla f_k\| \ll \|p_k\|$ in (5.41b), we conclude that

$$p_{k+1} \approx p_k,$$

so the new search direction will improve little (if at all) on the previous one. It follows that if the condition $\cos \theta_k \approx 0$ holds at some iteration k and if the subsequent step is small, a long sequence of unproductive iterates will follow.

The Polak–Ribière method behaves quite differently in these circumstances. If, as in the previous paragraph, the search direction p_k satisfies $\cos \theta_k \approx 0$ for some k , and if the subsequent step is small, it follows by substituting $\nabla f_k \approx \nabla f_{k+1}$ into (5.44) that $\beta_{k+1}^{\text{PR}} \approx 0$. From the formula (5.41b), we find that the new search direction p_{k+1} will be close to the steepest descent direction $-\nabla f_{k+1}$, and $\cos \theta_{k+1}$ will be close to 1. Therefore, Algorithm PR essentially performs a restart after it encounters a bad direction. The same argument can be applied to Algorithms PR+ and HS. For the FR-PR variant, defined by (5.48), we have noted already that $\beta_{k+1}^{\text{FR}} \approx 1$, and $\beta_{k+1}^{\text{PR}} \approx 0$. The formula (5.48) thus sets $\beta_{k+1} = \beta_{k+1}^{\text{PR}}$, as desired. Thus, the modification (5.48) seems to avoid the inefficiencies of the FR method, while falling back on this method for global convergence.

The undesirable behavior of the Fletcher–Reeves method predicted by the arguments given above can be observed in practice. For example, the paper [123] describes a problem with $n = 100$ in which $\cos \theta_k$ is of order 10^{-2} for hundreds of iterations and the steps $\|x_k - x_{k-1}\|$ are of order 10^{-2} . Algorithm FR requires thousands of iterations to solve this problem, while Algorithm PR requires just 37 iterations. In this example, the Fletcher–Reeves method performs much better if it is periodically restarted along the steepest descent direction, since each restart terminates the cycle of bad steps. In general, Algorithm FR should not be implemented without some kind of restart strategy.

GLOBAL CONVERGENCE

Unlike the linear conjugate gradient method, whose convergence properties are well understood and which is known to be optimal as described above, nonlinear conjugate gradient methods possess surprising, sometimes bizarre, convergence properties. We now present a few of the main results known for the Fletcher–Reeves and Polak–Ribière methods using practical line searches.

For the purposes of this section, we make the following (nonrestrictive) assumptions on the objective function.

Assumptions 5.1.

- (i) *The level set $\mathcal{L} := \{x \mid f(x) \leq f(x_0)\}$ is bounded;*
- (ii) *In some open neighborhood \mathcal{N} of \mathcal{L} , the objective function f is Lipschitz continuously differentiable.*

These assumptions imply that there is a constant $\bar{\gamma}$ such that

$$\|\nabla f(x)\| \leq \bar{\gamma}, \text{ for all } x \in \mathcal{L}. \quad (5.59)$$

Our main analytical tool in this section is Zoutendijk's theorem—Theorem 3.2 in Chapter 3. It states, that under Assumptions 5.1, any line search iteration of the form $x_{k+1} = x_k + \alpha_k p_k$, where p_k is a descent direction and α_k satisfies the Wolfe conditions (5.43) gives the limit

$$\sum_{k=0}^{\infty} \cos^2 \theta_k \|\nabla f_k\|^2 < \infty. \quad (5.60)$$

We can use this result to prove global convergence for algorithms that are periodically restarted by setting $\beta_k = 0$. If k_1, k_2 , and so on denote the iterations on which restarts occur, we have from (5.60) that

$$\sum_{k=k_1, k_2, \dots} \|\nabla f_k\|^2 < \infty. \quad (5.61)$$

If we allow no more than \bar{n} iterations between restarts, the sequence $\{k_j\}_{j=1}^{\infty}$ is infinite, and from (5.61) we have that $\lim_{j \rightarrow \infty} \|\nabla f_{k_j}\| = 0$. That is, a subsequence of gradients approaches zero, or equivalently,

$$\liminf_{k \rightarrow \infty} \|\nabla f_k\| = 0. \quad (5.62)$$

This result applies equally to restarted versions of all the algorithms discussed in this chapter.

It is more interesting, however, to study the global convergence of *unrestarted* conjugate gradient methods, because for large problems (say $n \geq 1000$) we expect to find a solution in many fewer than n iterations—the first point at which a regular restart would take place. Our study of large sequences of unrestarted conjugate gradient iterations reveals some surprising patterns in their behavior.

We can build on Lemma 5.6 and Zoutendijk's result (5.60) to prove a global convergence result for the Fletcher–Reeves method. While we cannot show that the limit of the sequence of gradients $\{\nabla f_k\}$ is zero, the following result shows that this sequence is not bounded away from zero.

Theorem 5.7 (Al-Baali [3]).

Suppose that Assumptions 5.1 hold, and that Algorithm 5.4 is implemented with a line search that satisfies the strong Wolfe conditions (5.43), with $0 < c_1 < c_2 < \frac{1}{2}$. Then

$$\liminf_{k \rightarrow \infty} \|\nabla f_k\| = 0. \quad (5.63)$$

PROOF. The proof is by contradiction. It assumes that the opposite of (5.63) holds, that is, there is a constant $\gamma > 0$ such that

$$\|\nabla f_k\| \geq \gamma, \quad (5.64)$$

for all k sufficiently large. By substituting the left inequality of (5.57) into Zoutendijk's condition (5.60), we obtain

$$\sum_{k=0}^{\infty} \frac{\|\nabla f_k\|^4}{\|p_k\|^2} < \infty. \quad (5.65)$$

By using (5.43b) and (5.53), we obtain that

$$|\nabla f_k^T p_{k-1}| \leq -c_2 \nabla f_{k-1}^T p_{k-1} \leq \frac{c_2}{1-c_2} \|\nabla f_{k-1}\|^2. \quad (5.66)$$

Thus, from (5.41b) and recalling the definition (5.41a) of β_k^{FR} we obtain

$$\begin{aligned} \|p_k\|^2 &\leq \|\nabla f_k\|^2 + 2\beta_k^{\text{FR}} |\nabla f_k^T p_{k-1}| + (\beta_k^{\text{FR}})^2 \|p_{k-1}\|^2 \\ &\leq \|\nabla f_k\|^2 + \frac{2c_2}{1-c_2} \beta_k^{\text{FR}} \|\nabla f_{k-1}\|^2 + (\beta_k^{\text{FR}})^2 \|p_{k-1}\|^2 \\ &= \left(\frac{1+c_2}{1-c_2} \right) \|\nabla f_k\|^2 + (\beta_k^{\text{FR}})^2 \|p_{k-1}\|^2. \end{aligned}$$

Applying this relation repeatedly, and defining $c_3 \stackrel{\text{def}}{=} (1+c_2)/(1-c_2) \geq 1$, we have

$$\begin{aligned} \|p_k\|^2 &\leq c_3 \|\nabla f_k\|^2 + (\beta_k^{\text{FR}})^2 (c_3 \|\nabla f_{k-1}\|^2 + (\beta_{k-1}^{\text{FR}})^2 (c_3 \|\nabla f_{k-2}\|^2 + \\ &\quad \dots + (\beta_1^{\text{FR}})^2 \|p_0\|^2)) \dots \\ &= c_3 \|\nabla f_k\|^4 \sum_{j=0}^k \|\nabla f_j\|^{-2}, \end{aligned} \quad (5.67)$$

where we used the facts that

$$(\beta_k^{\text{FR}})^2 (\beta_{k-1}^{\text{FR}})^2 \dots (\beta_{k-i}^{\text{FR}})^2 = \frac{\|\nabla f_k\|^4}{\|\nabla f_{k-i-1}\|^4}$$

and $p_0 = -\nabla f_0$. By using the bounds (5.59) and (5.64) in (5.67), we obtain

$$\|p_k\|^2 \leq \frac{c_3 \bar{\gamma}^4}{\gamma^2} k, \quad (5.68)$$

which implies that

$$\sum_{k=1}^{\infty} \frac{1}{\|p_k\|^2} \geq \gamma_4 \sum_{k=1}^{\infty} \frac{1}{k}, \quad (5.69)$$

for some positive constant γ_4 .

On the other hand, from (5.64) and (5.65), we have that

$$\sum_{k=1}^{\infty} \frac{1}{\|p_k\|^2} < \infty. \quad (5.70)$$

However, if we combine this inequality with (5.69), we obtain that $\sum_{k=1}^{\infty} 1/k < \infty$, which is not true. Hence, (5.64) does not hold, and the claim (5.63) is proved. \square

This global convergence result can be extended to any choice of β_k satisfying (5.47), and in particular to the FR-PR method given by (5.48).

In general, if we can show that there exist constants $c_4, c_5 > 0$ such that

$$\cos \theta_k \geq c_4 \frac{\|\nabla f_k\|}{\|p_k\|}, \quad \frac{\|\nabla f_k\|}{\|p_k\|} \geq c_5 > 0, \quad k = 1, 2, \dots,$$

it follows from (5.60) that

$$\lim_{k \rightarrow \infty} \|\nabla f_k\| = 0.$$

In fact, this result can be established for the Polak–Ribière method under the assumption that f is strongly convex and that an exact line search is used.

For general (nonconvex) functions, however, is it not possible to prove a result like Theorem 5.7 for Algorithm PR. This fact is unexpected, since the Polak–Ribière method performs better in practice than the Fletcher–Reeves method. The following surprising result shows that the Polak–Ribière method can cycle infinitely without approaching a solution point, even if an ideal line search is used. (By “ideal” we mean that line search returns a value α_k that is the first positive stationary point for the function $t(\alpha) = f(x_k + \alpha p_k)$.)

Theorem 5.8.

Consider the Polak–Ribière method method (5.44) with an ideal line search. There exists a twice continuously differentiable objective function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ and a starting point $x_0 \in \mathbb{R}^3$ such that the sequence of gradients $\{\|\nabla f_k\|\}$ is bounded away from zero.

The proof of this result, given in [253], is quite complex. It demonstrates the existence of the desired objective function without actually constructing this function explicitly. The result is interesting, since the step length assumed in the proof—the first stationary point—may be accepted by any of the practical line search algorithms currently in use. The proof

of Theorem 5.8 requires that some consecutive search directions become almost negatives of each other. In the case of ideal line searches, this happens only if $\beta_k < 0$, so the analysis suggests Algorithm PR+ (see (5.45)), in which we reset β_k to zero whenever it becomes negative. We mentioned earlier that a line search strategy based on a slight modification of the Wolfe conditions guarantees that all search directions generated by Algorithm PR+ are descent directions. Using these facts, it is possible to prove a global convergence result like Theorem 5.7 for Algorithm PR+. An attractive property of the formulae (5.49), (5.50) is that global convergence can be established without introducing any modification to a line search based on the Wolfe conditions.

NUMERICAL PERFORMANCE

Table 5.1 illustrates the performance of Algorithms FR, PR, and PR+ without restarts. For these tests, the parameters in the strong Wolfe conditions (5.43) were chosen to be $c_1 = 10^{-4}$ and $c_2 = 0.1$. The iterations were terminated when

$$\|\nabla f_k\|_\infty < 10^{-5}(1 + |f_k|).$$

If this condition was not satisfied after 10,000 iterations, we declare failure (indicated by a * in the table).

The final column, headed “mod,” indicates the number of iterations of Algorithm PR+ for which the adjustment (5.45) was needed to ensure that $\beta_k^{\text{PR}} \geq 0$. Algorithm FR on problem GENROS takes very short steps far from the solution that lead to tiny improvements in the objective function, and convergence was not achieved within the maximum number of iterations.

The Polak–Ribière algorithm, or its variation PR+, are not always more efficient than Algorithm FR, and it has the slight disadvantage of requiring one more vector of storage. Nevertheless, we recommend that users choose Algorithm PR, PR+ or FR-PR, or the methods based on (5.49) and (5.50).

Table 5.1 Iterations and function/gradient evaluations required by three nonlinear conjugate gradient methods on a set of test problems; see [123]

| Problem | n | Alg FR it/f-g | Alg PR it/f-g | Alg PR+ it/f-g | mod |
|----------|------|------------------|------------------|-------------------|-----|
| CALCVAR3 | 200 | 2808/5617 | 2631/5263 | 2631/5263 | 0 |
| GENROS | 500 | * | 1068/2151 | 1067/2149 | 1 |
| XPOWSING | 1000 | 533/1102 | 212/473 | 97/229 | 3 |
| TRIDIA1 | 1000 | 264/531 | 262/527 | 262/527 | 0 |
| MSQRT1 | 1000 | 422/849 | 113/231 | 113/231 | 0 |
| XPOWELL | 1000 | 568/1175 | 212/473 | 97/229 | 3 |
| TRIGON | 1000 | 231/467 | 40/92 | 40/92 | 0 |

NOTES AND REFERENCES

The conjugate gradient method was developed in the 1950s by Hestenes and Stiefel [168] as an alternative to factorization methods for finding solutions of symmetric positive definite systems. It was not until some years later, in one of the most important developments in sparse linear algebra, that this method came to be viewed as an iterative method that could give good approximate solutions to systems in many fewer than n steps. Our presentation of the linear conjugate gradient method follows that of Luenberger [195]. For a history of the development of the conjugate gradient and Lanczos methods see Golub and O’Leary [135].

Interestingly enough, the nonlinear conjugate gradient method of Fletcher and Reeves [107] was proposed after the linear conjugate gradient method had fallen out of favor, but several years before it was rediscovered as an iterative method for linear systems. The Polak–Ribière method was introduced in [237], and the example showing that it may fail to converge on nonconvex problems is given by Powell [253]. Restart procedures are discussed in Powell [248].

Hager and Zhang [161] report some of the best computational results obtained to date with a nonlinear CG method. Their implementation is based on formula (5.50) and uses a high-accuracy line search procedure. The results in Table 5.1 are taken from Gilbert and Nocedal [123]. This paper also describes a line search that guarantees that Algorithm PR+ always generates descent directions and proves global convergence.

Analysis due to Powell [245] provides further evidence of the inefficiency of the Fletcher–Reeves method using exact line searches. He shows that if the iterates enter a region in which the function is the two-dimensional quadratic

$$f(x) = \frac{1}{2}x^T x,$$

then the angle between the gradient ∇f_k and the search direction p_k stays constant. Since this angle can be arbitrarily close to 90° , the Fletcher–Reeves method can be slower than the steepest descent method. The Polak–Ribière method behaves quite differently in these circumstances: If a very small step is generated, the next search direction tends to the steepest descent direction, as argued above. This feature prevents a sequence of tiny steps.

The global convergence of nonlinear conjugate gradient methods has received much attention; see for example Al-Baali [3], Gilbert and Nocedal [123], Dai and Yuan [85], and Hager and Zhang [161]. For recent surveys on CG methods see Gould et al. [147] and Hager and Zhang [162].

Most of the theory on the rate of convergence of conjugate gradient methods assumes that the line search is exact. Crowder and Wolfe [82] show that the rate of convergence is linear, and show by constructing an example that Q-superlinear convergence is not achievable. Powell [245] studies the case in which the conjugate gradient method enters a region where the objective function is quadratic, and shows that either finite termination occurs or the rate of convergence is linear. Cohen [63] and Burmeister [45] prove n -step


quadratic convergence (5.51) for general objective functions. Ritter [265] shows that in fact, the rate is *superquadratic*, that is,


$$\|x_{k+n} - x^*\| = o(\|x_k - x^*\|^2).$$


Powell [251] gives a slightly better result and performs numerical tests on small problems to measure the rate observed in practice. He also summarizes rate-of-convergence results for asymptotically exact line searches, such as those obtained by Baptist and Stoer [11] and Stoer [282]. Even faster rates of convergence can be established (see Schuller [278], Ritter [265]), under the assumption that the search directions are uniformly linearly independent, but this assumption is hard to verify and does not often occur in practice.


Nemirovsky and Yudin [225] devote some attention to the *global efficiency* of the Fletcher–Reeves and Polak–Ribière methods with exact line searches. For this purpose they define a measure of “laboriousness” and an “optimal bound” for it among a certain class of iterations. They show that on strongly convex problems not only do the Fletcher–Reeves and Polak–Ribière methods fail to attain the optimal bound, but they may also be slower than the steepest descent method. Subsequently, Nesterov [225] presented an algorithm that attains this optimal bound. It is related to PARTAN, the method of parallel tangents (see, for example, Luenberger [195]). We feel that this approach is unlikely to be effective in practice, but no conclusive investigation has been carried out, to the best of our knowledge.


EXERCISES


 **5.1** Implement Algorithm 5.2 and use it to solve linear systems in which A is the Hilbert matrix, whose elements are $A_{i,j} = 1/(i + j - 1)$. Set the right-hand-side to $b = (1, 1, \dots, 1)^T$ and the initial point to $x_0 = 0$. Try dimensions $n = 5, 8, 12, 20$ and report the number of iterations required to reduce the residual below 10^{-6} .







 **5.2** Show that if the nonzero vectors p_0, p_1, \dots, p_l satisfy (5.5), where A is symmetric and positive definite, then these vectors are linearly independent. (This result implies that A has at most n conjugate directions.)

 **5.3** Verify the formula (5.7).

 **5.4** Show that if $f(x)$ is a strictly convex quadratic, then the function $h(\sigma) \stackrel{\text{def}}{=} f(x_0 + \sigma_0 p_0 + \dots + \sigma_{k-1} p_{k-1})$ also is a strictly convex quadratic in the variable $\sigma = (\sigma_0, \sigma_1, \dots, \sigma_{k-1})^T$.

 **5.5** Verify from the formulae (5.14) that (5.17) and (5.18) hold for $k = 1$.

 **5.6** Show that (5.24d) is equivalent to (5.14d).

-  **5.7** Let $\{\lambda_i, v_i\}$ $i = 1, 2, \dots, n$ be the eigenpairs of the symmetric matrix A . Show that the eigenvalues and eigenvectors of $[I + P_k(A)A]^T A [I + P_k(A)A]$ are $\lambda_i [1 + \lambda_i P_k(\lambda_i)]^2$ and v_i , respectively.
-  **5.8** Construct matrices with various eigenvalue distributions (clustered and non-clustered) and apply the CG method to them. Comment on whether the behavior can be explained from Theorem 5.5.
-  **5.9** Derive Algorithm 5.3 by applying the standard CG method in the variables \hat{x} and then transforming back into the original variables.
-  **5.10** Verify the modified conjugacy condition (5.40).
-  **5.11** Show that when applied to a quadratic function, with exact line searches, both the Polak–Ribière formula given by (5.44) and the Hestenes–Stiefel formula given by (5.46) reduce to the Fletcher–Reeves formula (5.41a).
-  **5.12** Prove that Lemma 5.6 holds for any choice of β_k satisfying $|\beta_k| \leq \beta_k^{\text{FR}}$.