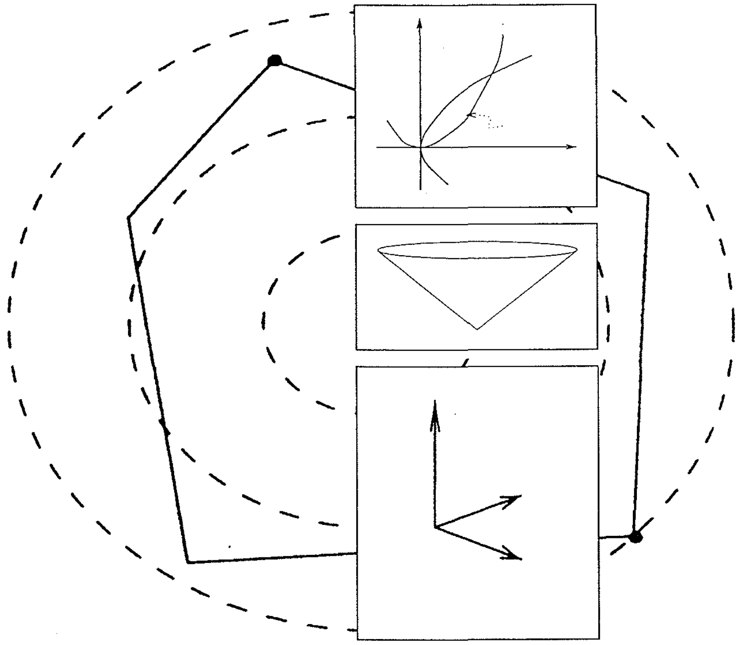
CHAPTER 2



Fundamentals of Unconstrained Optimization

In unconstrained optimization, we minimize an objective function that depends on real variables, with no restrictions at all on the values of these variables. The mathematical formulation is

$$\min_x f(x), \tag{2.1}$$

where $x \in \mathbb{R}^n$ is a real vector with $n \geq 1$ components and $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function.

Usually, we lack a global perspective on the function f. All we know are the values of f and maybe some of its derivatives at a set of points $x_0, x_1, x_2, \ldots$. Fortunately, our algorithms get to choose these points, and they try to do so in a way that identifies a solution reliably and without using too much computer time or storage. Often, the information about f does not come cheaply, so we usually prefer algorithms that do not call for this information unnecessarily.

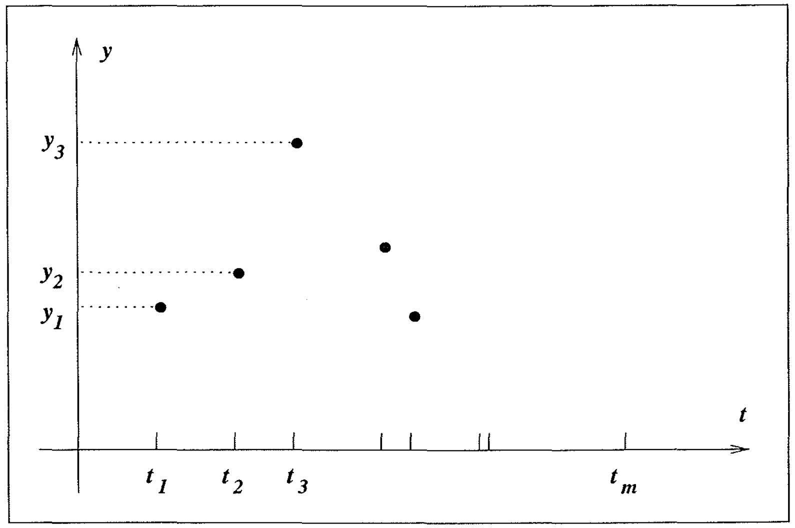


Figure 2.1 Least squares data fitting problem.

❑ **EXAMPLE 2.1**

Suppose that we are trying to find a curve that fits some experimental data. Figure 2.1 plots measurements $y_1, y_2, \ldots, y_m$ of a signal taken at times $t_1, t_2, \ldots, t_m$. From the data and our knowledge of the application, we deduce that the signal has exponential and oscillatory behavior of certain types, and we choose to model it by the function

$$\phi(t; x) = x_1 + x_2 e^{-(x_3-t)^2/x_4} + x_5 \cos(x_6 t).$$

The real numbers x_i, $i = 1, 2, \ldots, 6$, are the parameters of the model. We would like to choose them to make the model values $\phi(t_j; x)$ fit the observed data y_j as closely as possible. To state our objective as an optimization problem, we group the parameters x_i into a vector of unknowns $x = (x_1, x_2, \ldots, x_6)^T$, and define the residuals

$$r_j(x) = y_j - \phi(t_j; x), \qquad j = 1, \ldots, m, \tag{2.2}$$

which measure the discrepancy between the model and the observed data. Our estimate of x will be obtained by solving the problem

$$\min_{x \in \mathbb{R}^6} f(x) = r_1^2(x) + \cdots + r_m^2(x). \tag{2.3}$$

This is a *nonlinear least-squares problem*, a special case of unconstrained optimization. It illustrates that some objective functions can be expensive to evaluate even when the number of variables is small. Here we have $n = 6$, but if the number of measurements m is large (10^5, say), evaluation of $f(x)$ for a given parameter vector x is a significant computation. ❑

Suppose that for the data given in Figure 2.1 the optimal solution of (2.3) is approximately $x^* = (1.1, 0.01, 1.2, 1.5, 2.0, 1.5)$ and the corresponding function value is $f(x^*) = 0.34$. Because the optimal objective is nonzero, there must be discrepancies between the observed measurements y_j and the model predictions $\phi(t_j, x^*)$ for some (usually most) values of j—the model has not reproduced all the data points exactly. How, then, can we verify that x^* is indeed a minimizer of f? To answer this question, we need to define the term "solution" and explain how to recognize solutions. Only then can we discuss algorithms for unconstrained optimization problems.

2.1 WHAT IS A SOLUTION?

Generally, we would be happiest if we found a *global minimizer* of f, a point where the function attains its least value. A formal definition is

A point x^* is a *global minimizer* if $f(x^*) \leq f(x)$ for all x,

where x ranges over all of $\mathbb{R}^n$ (or at least over the domain of interest to the modeler). The global minimizer can be difficult to find, because our knowledge of f is usually only local. Since our algorithm does not visit many points (we hope!), we usually do not have a good picture of the overall shape of f, and we can never be sure that the function does not take a sharp dip in some region that has not been sampled by the algorithm. Most algorithms are able to find only a *local* minimizer, which is a point that achieves the smallest value of f in its neighborhood. Formally, we say:

A point x^* is a *local minimizer* if there is a neighborhood $\mathcal{N}$ of x^* such that $f(x^*) \leq f(x)$ for $x \in \mathcal{N}$.

(Recall that a neighborhood of x^* is simply an open set that contains x^*.) A point that satisfies this definition is sometimes called a *weak local minimizer*. This terminology distinguishes it from a strict local minimizer, which is the outright winner in its neighborhood. Formally,

A point x^* is a *strict local minimizer* (also called a *strong local minimizer*) if there is a neighborhood $\mathcal{N}$ of x^* such that $f(x^*) < f(x)$ for all $x \in \mathcal{N}$ with $x \neq x^*$.

For the constant function $f(x) = 2$, every point x is a weak local minimizer, while the function $f(x) = (x-2)^4$ has a strict local minimizer at $x = 2$.

A slightly more exotic type of local minimizer is defined as follows.

A point x^* is an *isolated local minimizer* if there is a neighborhood $\mathcal{N}$ of x^* such that x^* is the only local minimizer in $\mathcal{N}$.

Some strict local minimizers are not isolated, as illustrated by the function

$$f(x) = x^4 \cos(1/x) + 2x^4, \qquad f(0) = 0,$$

which is twice continuously differentiable and has a strict local minimizer at $x^* = 0$. However, there are strict local minimizers at many nearby points x_n, and we can label these points so that $x_n \to 0$ as $n \to \infty$.

While strict local minimizers are not always isolated, it is true that all isolated local minimizers are strict.

Figure 2.2 illustrates a function with many local minimizers. It is usually difficult to find the global minimizer for such functions, because algorithms tend to be "trapped" at the local minimizers. This example is by no means pathological. In optimization problems associated with the determination of molecular conformation, the potential function to be minimized may have millions of local minima.

Sometimes we have additional "global" knowledge about f that may help in identifying global minima. An important special case is that of convex functions, for which every local minimizer is also a global minimizer.

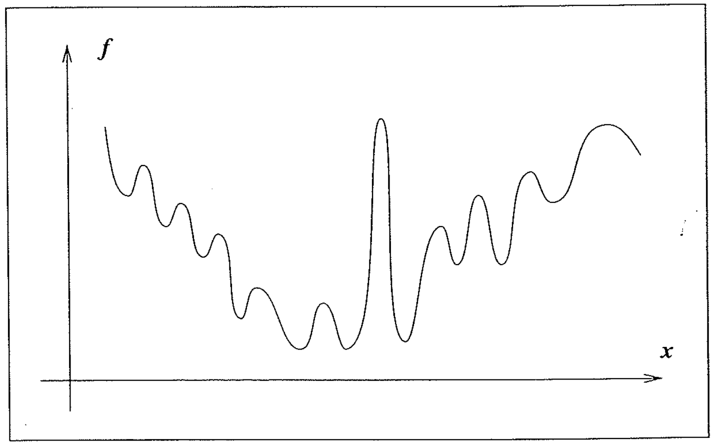


Figure 2.2 A difficult case for global minimization.

RECOGNIZING A LOCAL MINIMUM

From the definitions given above, it might seem that the only way to find out whether a point x^* is a local minimum is to examine all the points in its immediate vicinity, to make sure that none of them has a smaller function value. When the function f is *smooth*, however, there are much more efficient and practical ways to identify local minima. In particular, if f is twice continuously differentiable, we may be able to tell that x^* is a local minimizer (and possibly a strict local minimizer) by examining just the gradient $\nabla f(x^*)$ and the Hessian $\nabla^2 f(x^*)$.

The mathematical tool used to study minimizers of smooth functions is Taylor's theorem. Because this theorem is central to our analysis throughout the book, we state it now. Its proof can be found in any calculus textbook.

Theorem 2.1 (Taylor's Theorem).

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and that $p \in \mathbb{R}^n$. Then we have that

$$f(x+p) = f(x) + \nabla f(x+tp)^T p, \tag{2.4}$$

for some $t \in (0,1)$. Moreover, if f is twice continuously differentiable, we have that

$$\nabla f(x+p) = \nabla f(x) + \int_0^1 \nabla^2 f(x+tp)p\,dt, \tag{2.5}$$

and that

$$f(x+p) = f(x) + \nabla f(x)^T p + \tfrac{1}{2} p^T \nabla^2 f(x+tp) p, \tag{2.6}$$

for some $t \in (0,1)$.

Necessary conditions for optimality are derived by assuming that x^* is a local minimizer and then proving facts about $\nabla f(x^*)$ and $\nabla^2 f(x^*)$.

Theorem 2.2 (First-Order Necessary Conditions).

If x^* is a local minimizer and f is continuously differentiable in an open neighborhood of x^*, then $\nabla f(x^*) = 0$.

PROOF. Suppose for contradiction that $\nabla f(x^*) \neq 0$. Define the vector $p = -\nabla f(x^*)$ and note that $p^T \nabla f(x^*) = -\|\nabla f(x^*)\|^2 < 0$. Because ∇f is continuous near x^*, there is a scalar $T > 0$ such that

$$p^T \nabla f(x^* + tp) < 0, \qquad \text{for all } t \in [0, T].$$

For any $\bar{t} \in (0, T]$, we have by Taylor's theorem that

$$f(x^* + \bar{t}p) = f(x^*) + \bar{t}p^T \nabla f(x^* + tp), \qquad \text{for some } t \in (0, \bar{t}).$$

Therefore, $f(x^* + \bar{t}p) < f(x^*)$ for all $\bar{t} \in (0, T]$. We have found a direction leading away from x^* along which f decreases, so x^* is not a local minimizer, and we have a contradiction. □

We call x^* a *stationary point* if $\nabla f(x^*) = 0$. According to Theorem 2.2, any local minimizer must be a stationary point.

For the next result we recall that a matrix B is positive definite if $p^T Bp > 0$ for all $p \neq 0$, and positive semidefinite if $p^T Bp \geq 0$ for all p (see the Appendix).

Theorem 2.3 (Second-Order Necessary Conditions).

If x^* is a local minimizer of f and $\nabla^2 f$ is continuous in an open neighborhood of x^*, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.

PROOF. We know from Theorem 2.2 that $\nabla f(x^*) = 0$. For contradiction, assume that $\nabla^2 f(x^*)$ is not positive semidefinite. Then we can choose a vector p such that $p^T \nabla^2 f(x^*)p < 0$, and because $\nabla^2 f$ is continuous near x^*, there is a scalar $T > 0$ such that $p^T \nabla^2 f(x^* + tp)p < 0$ for all $t \in [0, T]$.

By doing a Taylor series expansion around x^*, we have for all $\bar{t} \in (0, T]$ and some $t \in (0, \bar{t})$ that

$$f(x^* + \bar{t}p) = f(x^*) + \bar{t}p^T \nabla f(x^*) + \tfrac{1}{2}\bar{t}^2 p^T \nabla^2 f(x^* + tp)p < f(x^*).$$

As in Theorem 2.2, we have found a direction from x^* along which f is decreasing, and so again, x^* is not a local minimizer. □

We now describe *sufficient conditions*, which are conditions on the derivatives of f at the point z^* that guarantee that x^* is a local minimizer.

Theorem 2.4 (Second-Order Sufficient Conditions).

Suppose that $\nabla^2 f$ is continuous in an open neighborhood of x^* and that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite. Then x^* is a strict local minimizer of f.

PROOF. Because the Hessian is continuous and positive definite at x^*, we can choose a radius $r > 0$ so that $\nabla^2 f(x)$ remains positive definite for all x in the open ball $\mathcal{D} = \{z \mid \|z - x^*\| < r\}$. Taking any nonzero vector p with $\|p\| < r$, we have $x^* + p \in \mathcal{D}$ and so

$$\begin{aligned} f(x^* + p) &= f(x^*) + p^T \nabla f(x^*) + \tfrac{1}{2} p^T \nabla^2 f(z)p \\ &= f(x^*) + \tfrac{1}{2} p^T \nabla^2 f(z)p, \end{aligned}$$

where $z = x^* + tp$ for some $t \in (0, 1)$. Since $z \in \mathcal{D}$, we have $p^T \nabla^2 f(z)p > 0$, and therefore $f(x^* + p) > f(x^*)$, giving the result. □

Note that the second-order sufficient conditions of Theorem 2.4 guarantee something stronger than the necessary conditions discussed earlier; namely, that the minimizer is a *strict* local minimizer. Note too that the second-order sufficient conditions are not necessary: A point x^* may be a strict local minimizer, and yet may fail to satisfy the sufficient conditions. A simple example is given by the function $f(x) = x^4$, for which the point $x^* = 0$ is a strict local minimizer at which the Hessian matrix vanishes (and is therefore not positive definite).

When the objective function is convex, local and global minimizers are simple to characterize.

Theorem 2.5.

When f is convex, any local minimizer x^* is a global minimizer of f. If in addition f is differentiable, then any stationary point x^* is a global minimizer of f.

PROOF. Suppose that x^* is a local but not a global minimizer. Then we can find a point $z \in \mathbb{R}^n$ with $f(z) < f(x^*)$. Consider the line segment that joins x^* to z, that is,

$$x = \lambda z + (1 - \lambda)x^*, \qquad \text{for some } \lambda \in (0, 1]. \tag{2.7}$$

By the convexity property for f, we have

$$f(x) \leq \lambda f(z) + (1 - \lambda) f(x^*) < f(x^*). \tag{2.8}$$

Any neighborhood $\mathcal{N}$ of x^* contains a piece of the line segment (2.7), so there will always be points $x \in \mathcal{N}$ at which (2.8) is satisfied. Hence, x^* is not a local minimizer.

For the second part of the theorem, suppose that x^* is not a global minimizer and choose z as above. Then, from convexity, we have

$$\begin{aligned} \nabla f(x^*)^T (z - x^*) &= \frac{d}{d\lambda} f(x^* + \lambda(z - x^*)) \,|_{\lambda=0} \qquad \text{(see the Appendix)} \\ &= \lim_{\lambda \downarrow 0} \frac{f(x^* + \lambda(z - x^*)) - f(x^*)}{\lambda} \\ &\leq \lim_{\lambda \downarrow 0} \frac{\lambda f(z) + (1 - \lambda) f(x^*) - f(x^*)}{\lambda} \\ &= f(z) - f(x^*) < 0. \end{aligned}$$

Therefore, $\nabla f(x^*) \neq 0$, and so x^* is not a stationary point. □

These results, which are based on elementary calculus, provide the foundations for unconstrained optimization algorithms. In one way or another, all algorithms seek a point where $\nabla f(\cdot)$ vanishes.

NONSMOOTH PROBLEMS

This book focuses on smooth functions, by which we generally mean functions whose second derivatives exist and are continuous. We note, however, that there are interesting problems in which the functions involved may be nonsmooth and even discontinuous. It is not possible in general to identify a minimizer of a general discontinuous function. If, however, the function consists of a few smooth pieces, with discontinuities between the pieces, it may be possible to find the minimizer by minimizing each smooth piece individually.

If the function is continuous everywhere but nondifferentiable at certain points, as in Figure 2.3, we can identify a solution by examing the *subgradient*, or *generalized gradient*, which is a generalization of the concept of gradient to the nonsmooth case. Nonsmooth optimization is beyond the scope of this book; we refer instead to Hiriart-Urruty and Lemaréchal [137] for an extensive discussion of theory. Here, we mention only that the minimization of a function such as the one illustrated in Figure 2.3 (which contains a jump discontinuity in the first derivative $f'(x)$ at the minimum) is difficult because the behavior of f is not predictable near the point of nonsmoothness. That is, we cannot be sure that information about f obtained at one point can be used to infer anything about f at neighboring points, because points of nondifferentiability may intervene. However, certain special nondifferentiable functions, such as functions of the form

$$f(x) = \|r(x)\|_1, \qquad f(x) = \|r(x)\|_\infty$$

(where $r(x)$ is the residual vector refined in (2.2)), can be solved with the help of special-purpose algorithms; see, for example, Fletcher [83, Chapter 14].

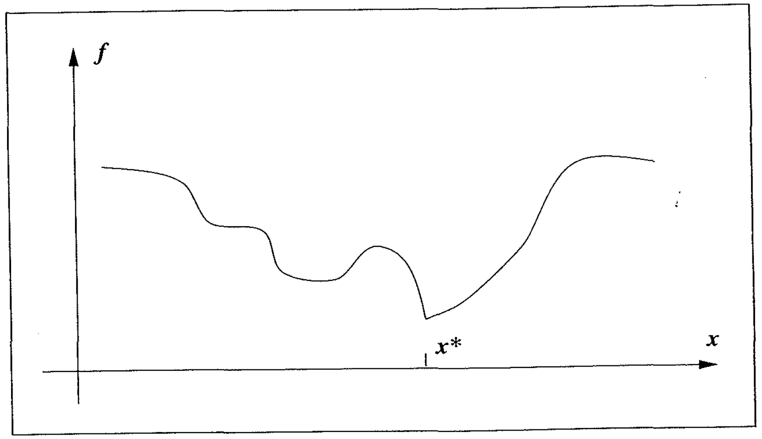


Figure 2.3 Nonsmooth function with minimum at a kink.

2.2 OVERVIEW OF ALGORITHMS

The last thirty years has seen the development of a powerful collection of algorithms for unconstrained optimization of smooth functions. We now give a broad description of their main properties, and we describe them in more detail in Chapters 3, 4, 5, 6, 8, and 9. All algorithms for unconstrained minimization require the user to supply a starting point, which we usually denote by x_0. The user with knowledge about the application and the data set may be in a good position to choose x_0 to be a reasonable estimate of the solution. Otherwise, the starting point must be chosen in some arbitrary manner.

Beginning at x_0, optimization algorithms generate a sequence of iterates $\{x_k\}_{k=0}^{\infty}$ that terminate when either no more progress can be made or when it seems that a solution point has been approximated with sufficient accuracy. In deciding how to move from one iterate x_k to the next, the algorithms use information about the function f at x_k, and possibly also information from earlier iterates $x_0, x_1, \ldots, x_{k-1}$. They use this information to find a new iterate x_{k+1} with a lower function value than x_k. (There exist *nonmonotone* algorithms that do not insist on a decrease in f at every step, but even these algorithms require f to be decreased after some prescribed number m of iterations. That is, they enforce $f(x_k) < f(x_{k-m})$.)

There are two fundamental strategies for moving from the current point x_k to a new iterate x_{k+1}. Most of the algorithms described in this book follow one of these approaches.

TWO STRATEGIES: LINE SEARCH AND TRUST REGION

In the *line search* strategy, the algorithm chooses a direction p_k and searches along this direction from the current iterate x_k for a new iterate with a lower function value. The distance to move along p_k can be found by approximately solving the following one-dimensional minimization problem to find a step length α:

$$\min_{\alpha>0} f(x_k + \alpha p_k). \tag{2.9}$$

By solving (2.9) exactly, we would derive the maximum benefit from the direction p_k, but an exact minimization is expensive and unnecessary. Instead, the line search algorithm generates a limited number of trial step lengths until it finds one that loosely approximates the minimum of (2.9). At the new point a new search direction and step length are computed, and the process is repeated.

In the second algorithmic strategy, known as *trust region*, the information gathered about f is used to construct a *model function* m_k whose behavior near the current point x_k is similar to that of the actual objective function f. Because the model m_k may not be a good approximation of f when x is far from x_k, we restrict the search for a minimizer of m_k to some region around x_k. In other words, we find the candidate step p by approximately solving the following subproblem:

$$\min_p m_k(x_k + p), \qquad \text{where } x_k + p \text{ lies inside the trust region.} \tag{2.10}$$

If the candidate solution does not produce a sufficient decrease in f, we conclude that the trust region is too large, and we shrink it and re-solve (2.10). Usually, the trust region is a ball defined by $\|p\|_2 \le \Delta$, where the scalar $\Delta > 0$ is called the trust-region radius. Elliptical and box-shaped trust regions may also be used.

The model m_k in (2.10) is usually defined to be a quadratic function of the form

$$m_k(x_k + p) = f_k + p^T \nabla f_k + \tfrac{1}{2} p^T B_k p, \tag{2.11}$$

where f_k, ∇f_k, and B_k are a scalar, vector, and matrix, respectively. As the notation indicates, f_k and ∇f_k are chosen to be the function and gradient values at the point x_k, so that m_k and f are in agreement to first order at the current iterate x_k. The matrix B_k is either the Hessian $\nabla^2 f_k$ or some approximation to it.

Suppose that the objective function is given by $f(x) = 10(x_2 - x_1^2)^2 + (1 - x_1)^2$. At the point $x_k = (0, 1)$ its gradient and Hessian are

$$\nabla f_k = \begin{bmatrix} -2 \\ 20 \end{bmatrix}, \qquad \nabla^2 f_k = \begin{bmatrix} -38 & 0 \\ 0 & 20 \end{bmatrix}.$$

The contour lines of the quadratic model (2.11) with $B_k = \nabla^2 f_k$ are depicted in Figure 2.4, which also illustrates the contours of the objective function f and the trust region. We have indicated contour lines where the model m_k has values 1 and 12. Note from Figure 2.4 that each time we decrease the size of the trust region after failure of a candidate iterate, the step from x_k to the new candidate will be shorter, and it usually points in a different direction from the previous candidate. The trust-region strategy differs in this respect from line search, which stays with a single search direction.

In a sense, the line search and trust-region approaches differ in the order in which they choose the *direction* and *distance* of the move to the next iterate. Line search starts by fixing the direction p_k and then identifying an appropriate distance, namely the step length α_k. In trust region, we first choose a maximum distance—the trust-region radius Δ_k—and then seek a direction and step that attain the best improvement possible subject to this distance constraint. If this step proves to be unsatisfactory, we reduce the distance measure Δ_k and try again.

The line search approach is discussed in more detail in Chapter 3. Chapter 4 discusses the trust-region strategy, including techniques for choosing and adjusting the size of the region and for computing approximate solutions to the trust-region problems (2.10). We now preview two major issues: choice of the search direction p_k in line search methods, and choice of the Hessian B_k in trust-region methods. These issues are closely related, as we now observe.

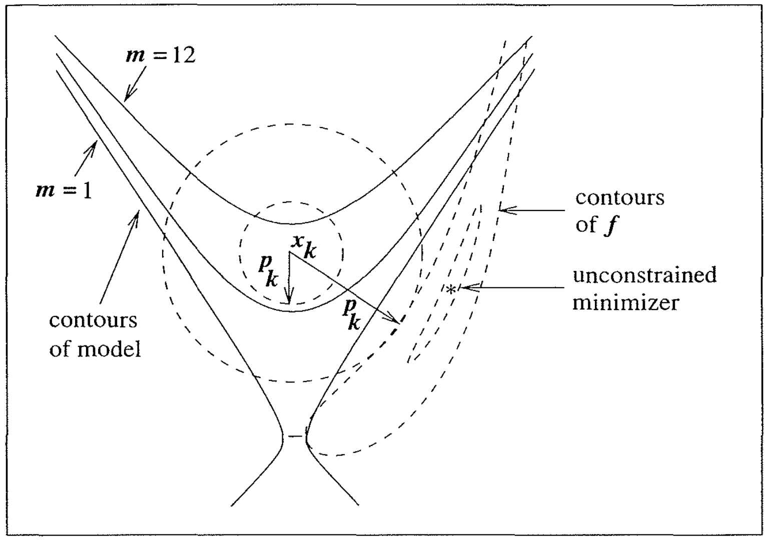


Figure 2.4 Two possible trust regions (circles) and their corresponding steps p_k. The solid lines are contours of the model function m_k.

SEARCH DIRECTIONS FOR LINE SEARCH METHODS

The steepest-descent direction $-\nabla f_k$ is the most obvious choice for search direction for a line search method. It is intuitive; among all the directions we could move from x_k, it is the one along which f decreases most rapidly. To verify this claim, we appeal again to Taylor's theorem (Theorem 2.1), which tells us that for any search direction p and step-length parameter α, we have

$$f(x_k + \alpha p) = f(x_k) + \alpha p^T \nabla f_k + \tfrac{1}{2}\alpha^2 p^T \nabla^2 f(x_k + tp) p, \qquad \text{for some } t \in (0, \alpha)$$

(see (2.6)). The rate of change in f along the direction p at x_k is simply the coefficient of α, namely, $p^T \nabla f_k$. Hence, the unit direction p of most rapid decrease is the solution to the problem

$$\min_p p^T \nabla f_k, \qquad \text{subject to } \|p\| = 1. \tag{2.12}$$

Since $p^T \nabla f_k = \|p\| \, \|\nabla f_k\| \cos\theta$, where θ is the angle between p and ∇f_k, we have from $\|p\| = 1$ that $p^T \nabla f_k = \|\nabla f_k\| \cos\theta$, so the objective in (2.12) is minimized when $\cos\theta$

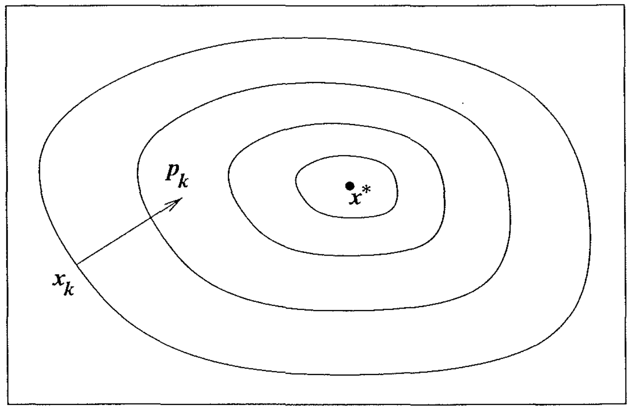


Figure 2.5 Steepest descent direction for a function of two variables.

takes on its minimum value of -1 at $\theta = \pi$ radians. In other words, the solution to (2.12) is

$$p = -\nabla f_k / \|\nabla f_k\|,$$

as claimed. As we show in Figure 2.5, this direction is orthogonal to the contours of the function.

The *steepest descent method* is a line search method that moves along $p_k = -\nabla f_k$ at every step. It can choose the step length α_k in a variety of ways, as we discuss in Chapter 3. One advantage of the steepest descent direction is that it requires calculation of the gradient ∇f_k but not of second derivatives. However, it can be excruciatingly slow on difficult problems.

Line search methods may use search directions other than the steepest descent direction. In general, any *descent* direction—one that makes an angle of strictly less than $\pi/2$ radians with $-\nabla f_k$—is guaranteed to produce a decrease in f, provided that the step length is sufficiently small (see Figure 2.6). We can verify this claim by using Taylor's theorem. From (2.6), we have that

$$f(x_k + \epsilon p_k) = f(x_k) + \epsilon p_k^T \nabla f_k + O(\epsilon^2).$$

When p_k is a downhill direction, the angle θ_k between p_k and ∇f_k has $\cos\theta_k < 0$, so that

$$p_k^T \nabla f_k = \|p_k\| \, \|\nabla f_k\| \cos\theta_k < 0.$$

It follows that $f(x_k + \epsilon p_k) < f(x_k)$ for all positive but sufficiently small values of ϵ.

Another important search direction—perhaps the most important one of all—is the *Newton direction*. This direction is derived from the second-order Taylor series

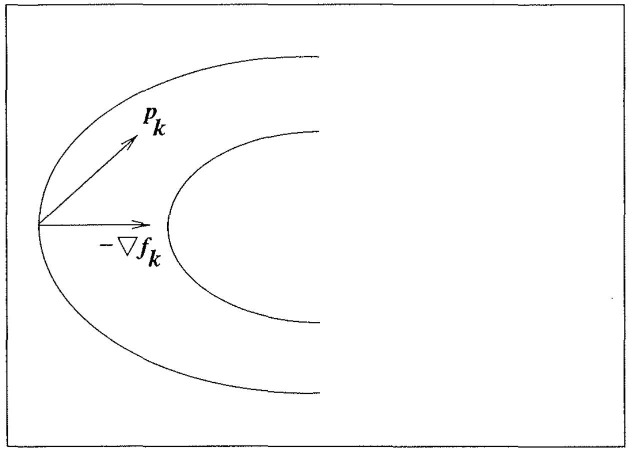


Figure 2.6 A downhill direction p_k

approximation to $f(x_k + p)$, which is

$$f(x_k + p) \approx f_k + p^T \nabla f_k + \tfrac{1}{2} p^T \nabla^2 f_k p \overset{\text{def}}{=} m_k(p). \tag{2.13}$$

Assuming for the moment that $\nabla^2 f_k$ is positive definite, we obtain the Newton direction by finding the vector p that minimizes $m_k(p)$. By simply setting the derivative of $m_k(p)$ to zero, we obtain the following explicit formula:

$$p_k^{\text{N}} = -\nabla^2 f_k^{-1} \nabla f_k. \tag{2.14}$$

The Newton direction is reliable when the difference between the true function $f(x_k + p)$ and its quadratic model $m_k(p)$ is not too large. By comparing (2.13) with (2.6), we see that the only difference between these functions is that the matrix $\nabla^2 f(x_k + tp)$ in the third term of the expansion has been replaced by $\nabla^2 f_k = \nabla^2 f(x_k)$. If $\nabla^2 f(\cdot)$ is sufficiently smooth, this difference introduces a perturbation of only $O(\|p\|^3)$ into the expansion, so that when $\|p\|$ is small, the approximation $f(x_k + p) \approx m_k(p)$ is very accurate indeed.

The Newton direction can be used in a line search method when $\nabla^2 f_k$ is positive definite, for in this case we have

$$\nabla f_k^T p_k^{\text{N}} = -p_k^{\text{N}T} \nabla^2 f_k p_k^{\text{N}} \le -\sigma_k \|p_k^{\text{N}}\|^2$$

for some $\sigma_k > 0$. Unless the gradient ∇f_k (and therefore the step p_k^{N}) is zero, we have that $\nabla f_k^T p_k^{\text{N}} < 0$, so the Newton direction is a descent direction. Unlike the steepest descent direction, there is a "natural" step length of 1 associated with the Newton direction. Most line search implementations of Newton's method use the unit step $\alpha = 1$ where possible and adjust this step length only when it does not produce a satisfactory reduction in the value of f.

When $\nabla^2 f_k$ is not positive definite, the Newton direction may not even be defined, since $\nabla^2 f_k^{-1}$ may not exist. Even when it *is* defined, it may not satisfy the descent property $\nabla f_k^T p_k^{\text{N}} < 0$, in which case it is unsuitable as a search direction. In these situations, line search methods modify the definition of p_k to make it satisfy the downhill condition while retaining the benefit of the second-order information contained in $\nabla^2 f_k$. We will describe these modifications in Chapter 6.

Methods that use the Newton direction have a fast rate of local convergence, typically quadratic. When a neighborhood of the solution is reached, convergence to high accuracy often occurs in just a few iterations. The main drawback of the Newton direction is the need for the Hessian $\nabla^2 f(x)$. Explicit computation of this matrix of second derivatives is sometimes, though not always, a cumbersome, error-prone, and expensive process.

Quasi-Newton search directions provide an attractive alternative in that they do not require computation of the Hessian and yet still attain a superlinear rate of convergence. In place of the true Hessian $\nabla^2 f_k$, they use an approximation B_k, which is updated after each step to take account of the additional knowledge gained during the step. The updates make use of the fact that changes in the gradient g provide information about the second derivative of f along the search direction. By using the expression (2.5) from our statement of Taylor's theorem, we have by adding and subtracting the term $\nabla^2 f(x)p$ that

$$\nabla f(x+p) = \nabla f(x) + \nabla^2 f(x)p + \int_0^1 \left[\nabla^2 f(x+tp) - \nabla^2 f(x)\right] p\, dt.$$

Because $\nabla f(\cdot)$ is continuous, the size of the final integral term is $o(\|p\|)$. By setting $x = x_k$ and $p = x_{k+1} - x_k$, we obtain

$$\nabla f_{k+1} = \nabla f_k + \nabla^2 f_{k+1}(x_{k+1} - x_k) + o(\|x_{k+1} - x_k\|).$$

When x_k and x_{k+1} lie in a region near the solution x^*, within which ∇f is positive definite, the final term in this expansion is eventually dominated by the $\nabla^2 f_k(x_{k+1} - x_k)$ term, and we can write

$$\nabla^2 f_{k+1}(x_{k+1} - x_k) \approx \nabla f_{k+1} - \nabla f_k. \tag{2.15}$$

We choose the new Hessian approximation B_{k+1} so that it mimics this property (2.15) of the true Hessian, that is, we require it to satisfy the following condition, known as the *secant equation*:

$$B_{k+1} s_k = y_k, \tag{2.16}$$

where

$$s_k = x_{k+1} - x_k, \qquad y_k = \nabla f_{k+1} - \nabla f_k.$$

Typically, we impose additional requirements on B_{k+1}, such as symmetry (motivated by symmetry of the exact Hessian), and a restriction that the difference between successive approximation B_k to B_{k+1} have low rank. The initial approximation B_0 must be chosen by the user.

Two of the most popular formulae for updating the Hessian approximation B_k are the *symmetric-rank-one* (SR1) formula, defined by

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}, \tag{2.17}$$

and the *BFGS formula*, named after its inventors, Broyden, Fletcher, Goldfarb, and Shanno, which is defined by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}. \tag{2.18}$$

Note that the difference between the matrices B_k and B_{k+1} is a rank-one matrix in the case of (2.17), and a rank-two matrix in the case of (2.18). Both updates satisfy the secant equation and both maintain symmetry. One can show that BFGS update (2.18) generates positive definite approximations whenever the initial approximation B_0 is positive definite and $s_k^T y_k > 0$. We discuss these issues further in Chapter 8.

The quasi-Newton search direction is given by using B_k in place of the exact Hessian in the formula (2.14), that is,

$$p_k = -B_k^{-1} \nabla f_k. \tag{2.19}$$

Some practical implementations of quasi-Newton methods avoid the need to factorize B_k at each iteration by updating the *inverse* of B_k, instead of B_k itself. In fact, the equivalent formula for (2.17) and (2.18), applied to the inverse approximation $H_k \overset{\text{def}}{=} B_k^{-1}$, is

$$H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho_k s_k s_k^T, \qquad \rho_k = \frac{1}{y_k^T s_k}. \tag{2.20}$$

Calculation of p_k can then be performed by using the formula $p_k = -H_k \nabla f_k$. This can be implemented as a matrix–vector multiplication, which is typically simpler than the factorization/back-substitution procedure that is needed to implement the formula (2.19).

Two variants of quasi-Newton methods designed to solve large problems—partially separable and limited-memory updating—are described in Chapter 9.

The last class of search directions we preview here is that generated by *nonlinear conjugate gradient methods.* They have the form

$$p_k = -\nabla f(x_k) + \beta_k p_{k-1},$$

where β_k is a scalar that ensures that p_k and p_{k-1} are *conjugate*—an important concept in the minimization of quadratic functions that will be defined in Chapter 5. Conjugate gradient methods were originally designed to solve systems of linear equations $Ax = b$, where the coefficient matrix A is symmetric and positive definite. The problem of solving this linear system is equivalent to the problem of minimizing the convex quadratic function defined by

$$\phi(x) = \tfrac{1}{2}x^T Ax + b^T x,$$

so it was natural to investigate extensions of these algorithms to more general types of unconstrained minimization problems. In general, nonlinear conjugate gradient directions are much more effective than the steepest descent direction and are almost as simple to compute. These methods do not attain the fast convergence rates of Newton or quasi-Newton methods, but they have the advantage of not requiring storage of matrices. An extensive discussion of nonlinear conjugate gradient methods is given in Chapter 5.

All of the search directions discussed so far can be used directly in a line search framework. They give rise to the steepest descent, Newton, quasi-Newton, and conjugate gradient line search methods. All except conjugate gradients have an analogue in the trust-region framework, as we now discuss.

MODELS FOR TRUST-REGION METHODS

If we set $B_k = 0$ in (2.11) and define the trust region using the Euclidean norm, the trust-region subproblem (2.10) becomes

$$\min_p \; f_k + p^T \nabla f_k \qquad \text{subject to } \|p\|_2 \le \Delta_k.$$

We can write the solution to this problem in closed form as

$$p_k = -\frac{\Delta_k \nabla f_k}{\|\nabla f_k\|}.$$

This is simply a steepest descent step in which the step length is determined by the trust-region radius; the trust-region and line search approaches are essentially the same in this case.

A more interesting trust-region algorithm is obtained by choosing B_k to be the exact Hessian $\nabla^2 f_k$ in the quadratic model (2.11). Because of the trust-region restriction $\|p\|_2 \le \Delta_k$, there is no need to do anything special when $\nabla^2 f_k$ is not positive definite, since the

subproblem (2.10) is guaranteed to have a solution p_k, as we see in Figure 2.4. The trust-region Newton method has proved to be highly effective in practice, as we discuss in Chapter 6.

If the matrix B_k in the quadratic model function m_k of (2.11) is defined by means of a quasi-Newton approximation, we obtain a trust-region quasi-Newton method.

SCALING

The performance of an algorithm may depend crucially on how the problem is formulated. One important issue in problem formulation is *scaling.* In unconstrained optimization, a problem is said to be *poorly scaled* if changes to x in a certain direction produce much larger variations in the value of f than do changes to x in another direction. A simple example is provided by the function $f(x) = 10^9 x_1^2 + x_2^2$, which is very sensitive to small changes in x_1 but not so sensitive to perturbations in x_2.

Poorly scaled functions arise, for example, in simulations of physical and chemical systems where different processes are taking place at very different rates. To be more specific, consider a chemical system in which four reactions occur. Associated with each reaction is a *rate constant* that describes the speed at which the reaction takes place. The optimization problem is to find values for these rate constants by observing the concentrations of each chemical in the system at different times. The four constants differ greatly in magnitude, since the reactions take place at vastly different speeds. Suppose we have the following rough estimates for the final values of the constants, each correct to within, say, an order of magnitude:

$$x_1 \approx 10^{-10}, \qquad x_2 \approx x_3 \approx 1, \qquad x_4 \approx 10^5.$$

Before solving this problem we could introduce a new variable z defined by

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 10^{-10} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 10^5 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix},$$

and then define and solve the optimization problem in terms of the new variable z. The optimal values of z will be within about an order of magnitude of 1, making the solution more balanced. This kind of scaling of the variables is known as *diagonal scaling.*

Scaling is performed (sometimes unintentionally) when the units used to represent variables are changed. During the modeling process, we may decide to change the units of some variables, say from meters to millimeters. If we do, the range of those variables and their size relative to the other variables will both change.

Some optimization algorithms, such as steepest descent, are sensitive to poor scaling, while others, such as Newton's method, are unaffected by it. Figure 2.7 shows the contours

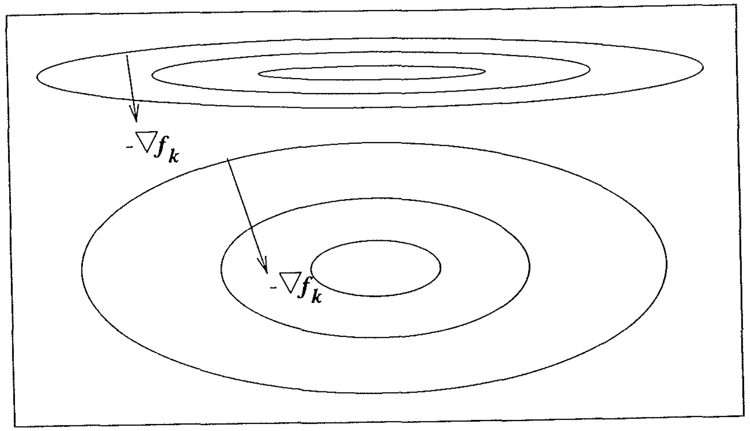


Figure 2.7 Poorly scaled and well-scaled problems, and performance of the steepest descent direction.

of two convex nearly quadratic functions, the first of which is poorly scaled, while the second is well scaled. For the poorly scaled problem, the one with highly elongated contours, the steepest descent direction (also shown on the graph) does not yield much reduction in the function, while for the well-scaled problem it performs much better. In both cases, Newton's method will produce a much better step, since the second-order quadratic model (m_k in (2.13)) happens to be a good approximation of f.

Algorithms that are not sensitive to scaling are preferable to those that are not, because they can handle poor problem formulations in a more robust fashion. In designing complete algorithms, we try to incorporate *scale invariance* into all aspects of the algorithm, including the line search or trust-region strategies and convergence tests. Generally speaking, it is easier to preserve scale invariance for line search algorithms than for trust-region algorithms.

RATES OF CONVERGENCE

One of the key measures of performance of an algorithm is its rate of convergence. We now define the terminology associated with different types of convergence, for reference in later chapters.

Let $\{x_k\}$ be a sequence in $\mathbb{R}^n$ that converges to x^*. We say that the convergence is *Q-linear* if there is a constant $r \in (0, 1)$ such that

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \le r, \quad \text{for all } k \text{ sufficiently large.} \tag{2.21}$$

This means that the distance to the solution x^* decreases at each iteration by at least a constant factor. For example, the sequence $1 + (0.5)^k$ converges Q-linearly to 1. The prefix "Q" stands for "quotient," because this type of convergence is defined in terms of the quotient of successive errors.

The convergence is said to be *Q-superlinear* if

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0.$$

For example, the sequence $1 + k^{-k}$ converges superlinearly to 1. (Prove this statement!) *Q-quadratic* convergence, an even more rapid convergence rate, is obtained if

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} \le M, \quad \text{for all } k \text{ sufficiently large,}$$

where M is a positive constant, not necessarily less than 1. An example is the sequence $1 + (0.5)^{2^k}$.

The speed of convergence depends on r and (more weakly) on M, whose values depend not only on the algorithm but also on the properties of the particular problem. Regardless of these values, however, a quadratically convergent sequence will always eventually converge faster than a linearly convergent sequence.

Obviously, any sequence that converges Q-quadratically also converges Q-superlinearly, and any sequence that converges Q-superlinearly also converges Q-linearly. We can also define higher rates of convergence (cubic, quartic, and so on), but these are less interesting in practical terms. In general, we say that the Q-order of convergence is p (with $p > 1$) if there is a positive constant M such that

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} \le M, \quad \text{for all } k \text{ sufficiently large.}$$

Quasi-Newton methods typically converge Q-superlinearly, whereas Newton's method converges Q-quadratically. In contrast, steepest descent algorithms converge only at a Q-linear rate, and when the problem is ill-conditioned the convergence constant r in (2.21) is close to 1.

Throughout the book we will normally omit the letter Q and simply talk about superlinear convergence, quadratic convergence, etc.

R-RATES OF CONVERGENCE

A slightly weaker form of convergence, characterized by the prefix "R" (for "root"), is concerned with the overall rate of decrease in the error, rather than the decrease over a single step of the algorithm. We say that convergence is *R-linear* if there is a sequence of nonnegative scalars $\{\nu_k\}$ such that

$$\|x_k - x^*\| \le \nu_k \text{ for all } k, \text{ and } \{\nu_k\} \text{ converges Q-linearly to zero.}$$

The sequence $\{\|x_k - x^*\|\}$ is said to be *dominated* by $\{\nu_k\}$. For instance, the sequence

$$x_k = \begin{cases} 1 + (0.5)^k, & k \text{ even}, \\ 1, & k \text{ odd}, \end{cases} \tag{2.22}$$

(the first few iterates are 2, 1, 1.25, 1, 1.03125, 1, ...) converges R-linearly to 1, because it is dominated by the sequence $1+(0.5)^k$, which converges Q-linearly. Likewise, we say that $\{x_k\}$ converges R-superlinearly to x^* if $\{\|x_k - x^*\|\}$ is dominated by a Q-superlinear sequence, and $\{x_k\}$ converges R-quadratically to x^* if $\{\|x_k - x^*\|\}$ is dominated by a Q-quadratic sequence.

Note that in the R-linear sequence (2.22), the error actually increases at every second iteration! Such behavior occurs even in sequences whose R-rate of convergence is arbitrarily high, but it cannot occur for Q-linear sequences, which insist on a decrease at every step k, for k sufficiently large.

Most convergence analyses of optimization algorithms are concerned with Q-convergence.

NOTES AND REFERENCES

For an extensive discussion of convergence rates see Ortega and Rheinboldt [185].

EXERCISES

2.1 Compute the gradient $\nabla f(x)$ and Hessian $\nabla^2 f(x)$ of the Rosenbrock function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \tag{2.23}$$

Show that $x^* = (1,1)^T$ is the only local minimizer of this function, and that the Hessian matrix at that point is positive definite.

2.2 Show that the function $f(x) = 8x_1 + 12x_2 + x_1^2 - 2x_2^2$ has only one stationary point, and that it is neither a maximum or minimum, but a saddle point. Sketch the contour lines of f.

2.3 Let a be a given n-vector, and A be a given $n \times n$ symmetric matrix. Compute the gradient and Hessian of $f_1(x) = a^T x$ and $f_2(x) = x^T A x$.

2.4 Write the second-order Taylor expansion (2.6) for the function $\cos(1/x)$ around a nonzero point x, and the third-order Taylor expansion of $\cos(x)$ around any point x. Evaluate the second expansion for the specific case of $x = 1$.

2.5 Consider the function $f : \mathbb{R}^2 \to \mathbb{R}$ defined by $f(x) = \|x\|^2$. Show that the sequence of iterates $\{x_k\}$ defined by

$$x_k = \left(1 + \frac{1}{2^k}\right) \begin{bmatrix} \cos k \\ \sin k \end{bmatrix}$$

satisfies $f(x_{k+1}) < f(x_k)$ for $k = 0, 1, 2, \ldots$. Show that every point on the unit circle $\{x \mid \|x\|^2 = 1\}$ is a limit point for $\{x_k\}$. Hint: Every value $\theta \in [0, 2\pi]$ is a limit point of the subsequence $\{\xi_k\}$ defined by

$$\xi_k = k (\text{mod } 2\pi) = k - 2\pi \left\lfloor \frac{k}{2\pi} \right\rfloor,$$

where the operator $\lfloor \cdot \rfloor$ denotes rounding down to the next integer.

2.6 Prove that all isolated local minimizers are strict. (Hint: Take an isolated local minimizer x^* and a neighborhood $\mathcal{N}$. Show that for any $x \in \mathcal{N}$, $x \neq x^*$ we must have $f(x) > f(x^*)$.)

2.7 Suppose that f is a convex function. Show that the set of global minimizers of f is a convex set.

2.8 Consider the function $f(x_1, x_2) = \left(x_1 + x_2^2\right)^2$. At the point $x^T = (1, 0)$ we consider the search direction $p^T = (-1, 1)$. Show that p is a descent direction and find all minimizers of the problem (2.9).

2.9 Suppose that $\tilde{f}(z) = f(x)$, where $x = Sz + s$ for some $S \in \mathbb{R}^{n \times n}$ and $s \in \mathbb{R}^n$. Show that

$$\nabla \tilde{f}(z) = S^T \nabla f(x), \qquad \nabla^2 \tilde{f}(z) = S^T \nabla^2 f(x) S.$$

(Hint: Use the chain rule to express $d\tilde{f}/dz_j$ in terms of df/dx_i and dx_i/dz_j for all $i, j = 1, 2, \ldots, n$.)

2.10 Show that the symmetric rank-one update (2.17) and the BFGS update (2.18) are scale-invariant if the initial Hessian approximations B_0 are chosen appropriately. That is, using the notation of the previous exercise, show that if these methods are applied to $f(x)$ starting from $x_0 = Sz_0 + s$ with initial Hessian B_0, and to $\tilde{f}(z)$ starting from z_0 with initial Hessian $S^T B_0 S$, then all iterates are related by $x_k = Sz_k + s$. (Assume for simplicity that the methods take unit step lengths.)

2.11 Suppose that a function f of two variables is poorly scaled at the solution x^*. Write two Taylor expansions of f around x^*—one along each coordinate direction—and use them to show that the Hessian $\nabla^2 f(x^*)$ is ill-conditioned.

✎ **2.12** Show that the sequence $x_k = 1/k$ is not Q-linearly convergent, though it does converge to zero. (This is called *sublinear convergence.*)

✎ **2.13** Show that the sequence $x_k = 1 + (0.5)^{2^k}$ is Q–quadratically convergent to 1.

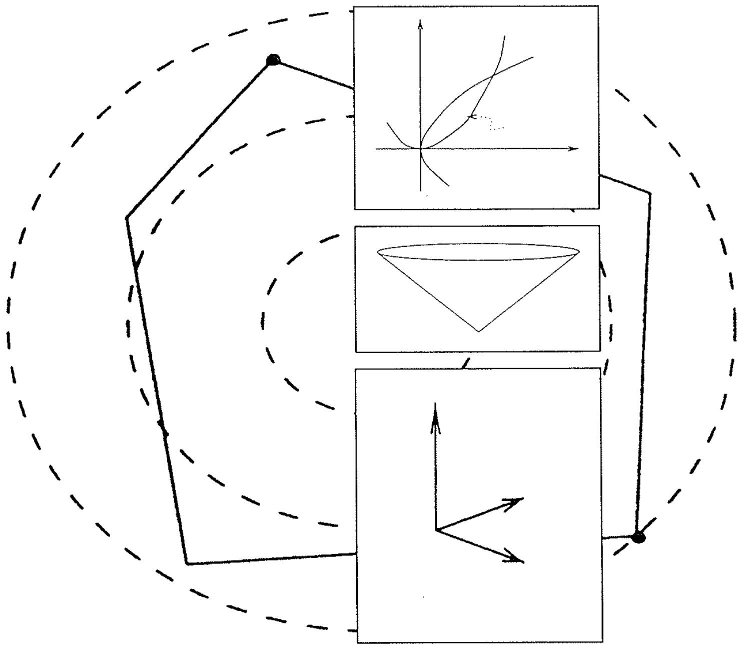
✎ **2.14** Does the sequence $1/(k!)$ converge Q-superlinearly? Q-quadratically?

✎ * **2.15** Consider the sequence $\{x_k\}$ defined by

$$x_k = \begin{cases} \left(\frac{1}{4}\right)^{2^k}, & k \text{ even}, \\ (x_{k-1})/k, & k \text{ odd}. \end{cases}$$

Is this sequence Q-superlinearly convergent? Q-quadratically convergent? R-quadratically convergent?

CHAPTER 3



Line Search Methods

Each iteration of a line search method computes a search direction p_k and then decides how far to move along that direction. The iteration is given by

$$x_{k+1} = x_k + \alpha_k p_k, \tag{3.1}$$

where the positive scalar α_k is called the *step length*. The success of a line search method depends on effective choices of both the direction p_k and the step length α_k.

Most line search algorithms require p_k to be a *descent direction*—one for which $p_k^T \nabla f_k < 0$—because this property guarantees that the function f can be reduced along this direction, as discussed in the previous chapter. Moreover, the search direction often has the form

$$p_k = -B_k^{-1} \nabla f_k, \tag{3.2}$$

where B_k is a symmetric and nonsingular matrix. In the steepest descent method B_k is simply the identity matrix I, while in Newton's method B_k is the exact Hessian $\nabla^2 f(x_k)$. In quasi-Newton methods, B_k is an approximation to the Hessian that is updated at every

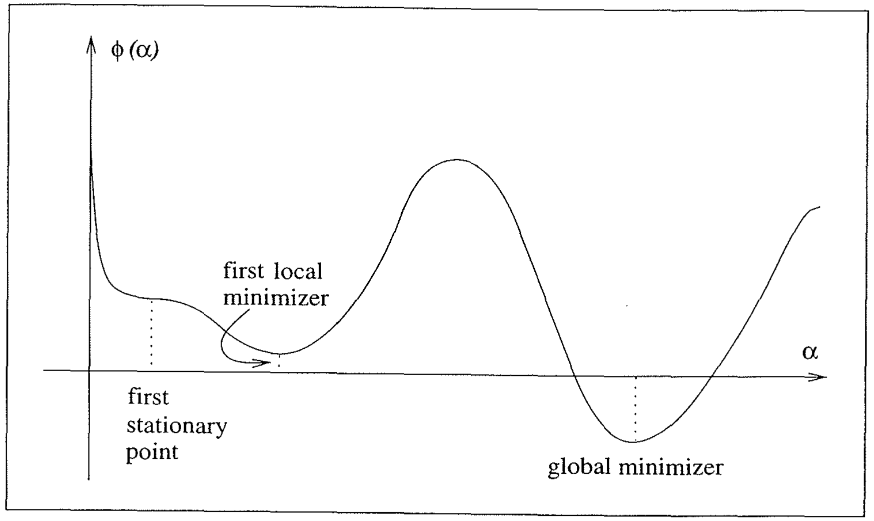


Figure 3.1 The ideal step length is the global minimizer.

iteration by means of a low-rank formula. When p_k is defined by (3.2) and B_k is positive definite, we have

$$p_k^T \nabla f_k = -\nabla f_k^T B_k^{-1} \nabla f_k < 0,$$

and therefore p_k is a descent direction.

In the next chapters we study how to choose the matrix B_k, or more generally, how to compute the search direction. We now give careful consideration to the choice of the step-length parameter α_k.

3.1 STEP LENGTH

In computing the step length α_k, we face a tradeoff. We would like to choose α_k to give a substantial reduction of f, but at the same time, we do not want to spend too much time making the choice. The ideal choice would be the global minimizer of the univariate function $\phi(\cdot)$ defined by

$$\phi(\alpha) = f(x_k + \alpha p_k), \quad \alpha > 0, \tag{3.3}$$

but in general, it is too expensive to identify this value (see Figure 3.1). To find even a local minimizer of ϕ to moderate precision generally requires too many evaluations of the objective function f and possibly the gradient ∇f. More practical strategies perform an *inexact* line search to identify a step length that achieves adequate reductions in f at minimal cost.

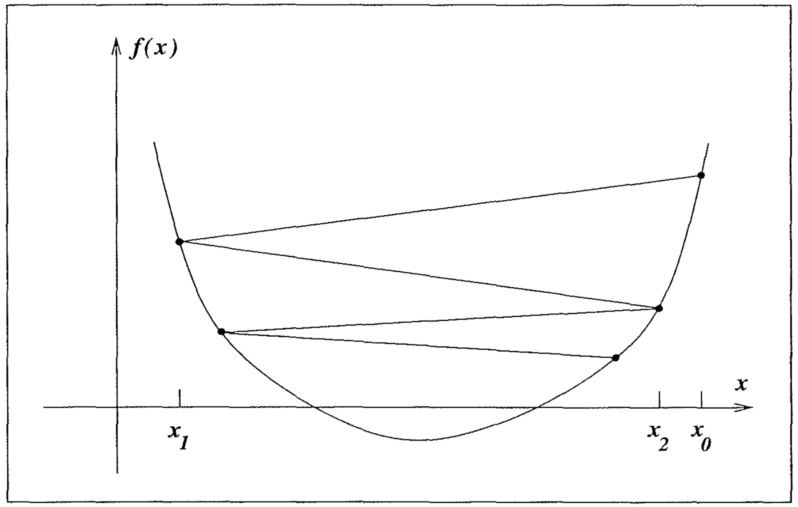


Figure 3.2 Insufficient reduction in f.

Typical line search algorithms try out a sequence of candidate values for α, stopping to accept one of these values when certain conditions are satisfied. The line search is done in two stages: A bracketing phase finds an interval containing desirable step lengths, and a bisection or interpolation phase computes a good step length within this interval. Sophisticated line search algorithms can be quite complicated, so we defer a full description until the end of this chapter. We now discuss various termination conditions for the line search algorithm and show that effective step lengths need not lie near minimizers of the univariate function $\phi(\alpha)$ defined in (3.3).

A simple condition we could impose on α_k is that it provide a reduction in f, i.e., $f(x_k + \alpha_k p_k) < f(x_k)$. That this is not appropriate is illustrated in Figure 3.2, where the minimum is $f^* = -1$, but the sequence of function values $\{5/k\}$, $k = 0, 1, \ldots$, converges to zero. The difficulty is that we do not have sufficient reduction in f, a concept we discuss next.

THE WOLFE CONDITIONS

A popular inexact line search condition stipulates that α_k should first of all give *sufficient decrease* in the objective function f, as measured by the following inequality:

$$f(x_k + \alpha p_k) \le f(x_k) + c_1 \alpha \nabla f_k^T p_k, \tag{3.4}$$

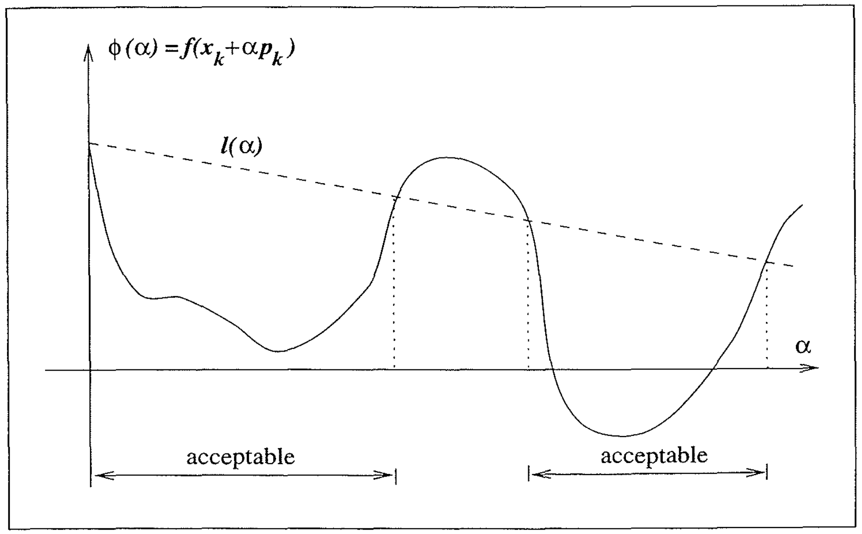


Figure 3.3 Sufficient decrease condition.

for some constant $c_1 \in (0, 1)$. In other words, the reduction in f should be proportional to both the step length α_k and the directional derivative $\nabla f_k^T p_k$. Inequality (3.4) is sometimes called the *Armijo condition.*

The sufficient decrease condition is illustrated in Figure 3.3. The right-hand-side of (3.4), which is a linear function, can be denoted by $l(\alpha)$. The function $l(\cdot)$ has negative slope $c_1 \nabla f_k^T p_k$, but because $c_1 \in (0, 1)$, it lies above the graph of ϕ for small positive values of α. The sufficient decrease condition states that α is acceptable only if $\phi(\alpha) \leq l(\alpha)$. The intervals on which this condition is satisfied are shown in Figure 3.3. In practice, c_1 is chosen to be quite small, say $c_1 = 10^{-4}$.

The sufficient decrease condition is not enough by itself to ensure that the algorithm makes reasonable progress, because as we see from Figure 3.3, it is satisfied for all sufficiently small values of α. To rule out unacceptably short steps we introduce a second requirement, called the *curvature condition,* which requires α_k to satisfy

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k, \tag{3.5}$$

for some constant $c_2 \in (c_1, 1)$, where c_1 is the constant from (3.4). Note that the left-hand-side is simply the derivative $\phi'(\alpha_k)$, so the curvature condition ensures that the slope of $\phi(\alpha_k)$ is greater than c_2 times the gradient $\phi'(0)$. This makes sense because if the slope $\phi'(\alpha)$ is strongly negative, we have an indication that we can reduce f significantly by moving further along the chosen direction. On the other hand, if the slope is only slightly negative or even positive, it is a sign that we cannot expect much more decrease in f in this direction,

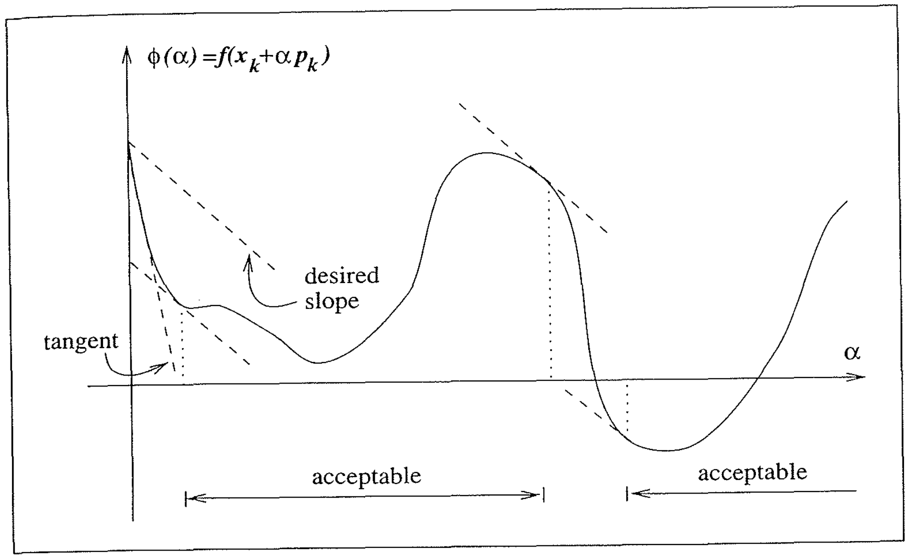


Figure 3.4 The curvature condition.

so it might make sense to terminate the line search. The curvature condition is illustrated in Figure 3.4. Typical values of c_2 are 0.9 when the search direction p_k is chosen by a Newton or quasi-Newton method, and 0.1 when p_k is obtained from a nonlinear conjugate gradient method.

The sufficient decrease and curvature conditions are known collectively as the *Wolfe conditions.* We illustrate them in Figure 3.5 and restate them here for future reference:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, \tag{3.6a}$$

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k, \tag{3.6b}$$

with $0 < c_1 < c_2 < 1$.

A step length may satisfy the Wolfe conditions without being particularly close to a minimizer of ϕ, as we show in Figure 3.5. We can, however, modify the curvature condition to force α_k to lie in at least a broad neighborhood of a local minimizer or stationary point of ϕ. The *strong Wolfe conditions* require α_k to satisfy

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, \tag{3.7a}$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq c_2 |\nabla f_k^T p_k|, \tag{3.7b}$$

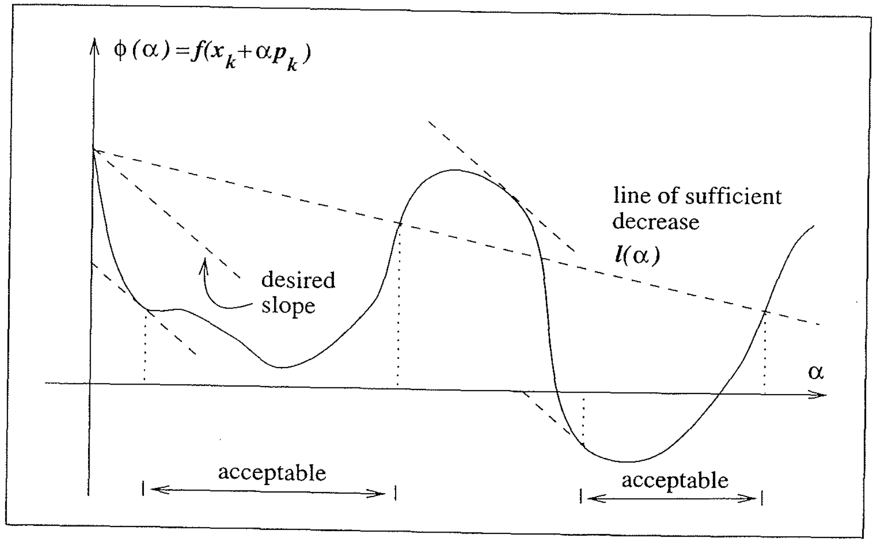


Figure 3.5 Step lengths satisfying the Wolfe conditions.

with $0 < c_1 < c_2 < 1$. The only difference with the Wolfe conditions is that we no longer allow the derivative $\phi'(\alpha_k)$ to be too positive. Hence, we exclude points that are far from stationary points of ϕ.

It is not difficult to prove that there exist step lengths that satisfy the Wolfe conditions for every function f that is smooth and bounded below.

Lemma 3.1.

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable. Let p_k be a descent direction at x_k, and assume that f is bounded below along the ray $\{x_k + \alpha p_k | \alpha > 0\}$. Then if $0 < c_1 < c_2 < 1$, there exist intervals of step lengths satisfying the Wolfe conditions (3.6) and the strong Wolfe conditions (3.7).

PROOF. Since $\phi(\alpha) = f(x_k + \alpha p_k)$ is bounded below for all $\alpha > 0$ and since $0 < c_1 < 1$, the line $l(\alpha) = f(x_k) + \alpha c_1 \nabla f_k^T p_k$ must intersect the graph of ϕ at least once. Let $\alpha' > 0$ be the smallest intersecting value of α, that is,

$$f(x_k + \alpha' p_k) = f(x_k) + \alpha' c_1 \nabla f_k^T p_k. \tag{3.8}$$

The sufficient decrease condition (3.6a) clearly holds for all step lengths less than α'.

By the mean value theorem, there exists $\alpha'' \in (0, \alpha')$ such that

$$f(x_k + \alpha' p_k) - f(x_k) = \alpha' \nabla f(x_k + \alpha'' p_k)^T p_k. \tag{3.9}$$

By combining (3.8) and (3.9), we obtain

$$\nabla f(x_k + \alpha'' p_k)^T p_k = c_1 \nabla f_k^T p_k > c_2 \nabla f_k^T p_k, \tag{3.10}$$

since $c_1 < c_2$ and $\nabla f_k^T p_k < 0$. Therefore, α'' satisfies the Wolfe conditions (3.6), and the inequalities hold strictly in both (3.6a) and (3.6b). Hence, by our smoothness assumption on f, there is an interval around α'' for which the Wolfe conditions hold. Moreover, since the term in the left-hand side of (3.10) is negative, the strong Wolfe conditions (3.7) hold in the same interval. □

The Wolfe conditions are scale-invariant in a broad sense: Multiplying the objective function by a constant or making an affine change of variables does not alter them. They can be used in most line search methods, and are particularly important in the implementation of quasi-Newton methods, as we see in Chapter 8.

THE GOLDSTEIN CONDITIONS

Like the Wolfe conditions, the *Goldstein conditions* also ensure that the step length α achieves sufficient decrease while preventing α from being too small. The Goldstein conditions can also be stated as a pair of inequalities, in the following way:

$$f(x_k) + (1-c)\alpha_k \nabla f_k^T p_k \le f(x_k + \alpha_k p_k) \le f(x_k) + c\alpha_k \nabla f_k^T p_k, \tag{3.11}$$

with $0 < c < \frac{1}{2}$. The second inequality is the sufficient decrease condition (3.4), whereas the first inequality is introduced to control the step length from below; see Figure 3.6

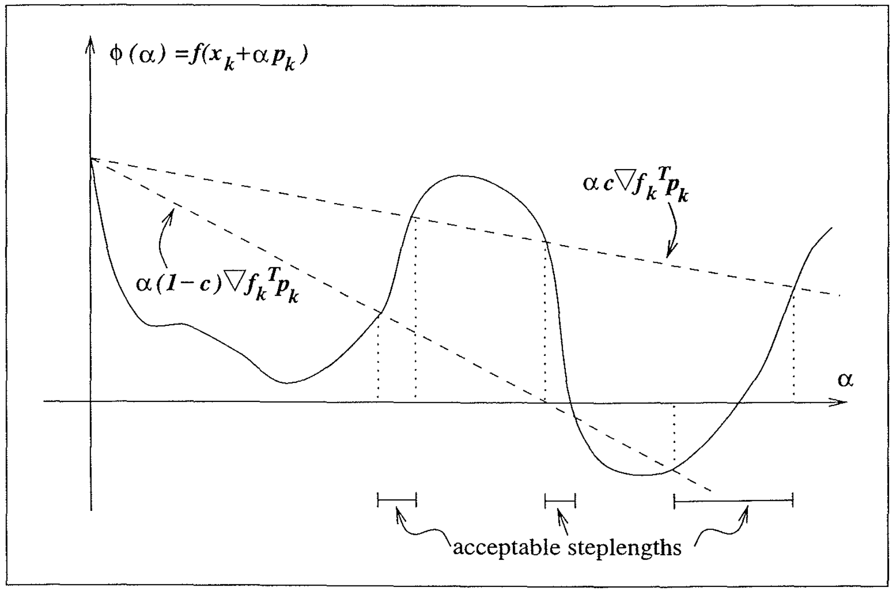
A disadvantage of the Goldstein conditions vis-à-vis the Wolfe conditions is that the first inequality in (3.11) may exclude all minimizers of ϕ. However, the Goldstein and Wolfe conditions have much in common, and their convergence theories are quite similar. The Goldstein conditions are often used in Newton-type methods but are not well suited for quasi-Newton methods that maintain a positive definite Hessian approximation.

SUFFICIENT DECREASE AND BACKTRACKING

We have mentioned that the sufficient decrease condition (3.6a) alone is not sufficient to ensure that the algorithm makes reasonable progress along the given search direction. However, if the line search algorithm chooses its candidate step lengths appropriately, by using a so-called *backtracking* approach, we can dispense with the extra condition (3.6b) and use just the sufficient decrease condition to terminate the line search procedure. In its most basic form, backtracking proceeds as follows.

Procedure 3.1 (Backtracking Line Search).

```
Choose $\bar{\alpha} > 0$, $\rho, c \in (0, 1)$; set $\alpha \leftarrow \bar{\alpha}$;
repeat until $f(x_k + \alpha p_k) \le f(x_k) + c\alpha \nabla f_k^T p_k$
```



$\phi(\alpha) = f(x_k + \alpha p_k)$
$\alpha c \nabla f_k^T p_k$
$\alpha(1-c)\nabla f_k^T p_k$
α
acceptable steplengths

Figure 3.6 The Goldstein conditions.

```
    $\alpha \leftarrow \rho\alpha$;
end (repeat)
Terminate with $\alpha_k = \alpha$.
```

In this procedure, the initial step length $\bar{\alpha}$ is chosen to be 1 in Newton and quasi-Newton methods, but can have different values in other algorithms such as steepest descent or conjugate gradient. An acceptable step length will be found after a finite number of trials because α_k will eventually become small enough that the sufficient decrease condition holds (see Figure 3.3). In practice, the contraction factor ρ is often allowed to vary at each iteration of the line search. For example, it can be chosen by safeguarded interpolation, as we describe later. We need ensure only that at each iteration we have $\rho \in [\rho_{\text{lo}}, \rho_{\text{hi}}]$, for some fixed constants $0 < \rho_{\text{lo}} < \rho_{\text{hi}} < 1$.

The backtracking approach ensures either that the selected step length α_k is some fixed value (the initial choice $\bar{\alpha}$), or else that it is short enough to satisfy the sufficient decrease condition but not *too* short. The latter claim holds because the accepted value α_k is within striking distance of the previous trial value, α_k/ρ, which was rejected for violating the sufficient decrease condition, that is, for being too long.

This simple and popular strategy for terminating a line search is well suited for Newton methods (see Chapter 6) but is less appropriate for quasi-Newton and conjugate gradient methods.

3.2 CONVERGENCE OF LINE SEARCH METHODS

To obtain global convergence, we must not only have well-chosen step lengths but also well-chosen search directions p_k. We discuss requirements on the search direction in this section, focusing on one key property: the angle θ_k between p_k and the steepest descent direction $-\nabla f_k$, defined by

$$\cos\theta_k = \frac{-\nabla f_k^T p_k}{\|\nabla f_k\| \, \|p_k\|}. \tag{3.12}$$

The following theorem, due to Zoutendijk, has far-reaching consequences. It shows, for example, that the steepest descent method is globally convergent. For other algorithms it describes how far p_k can deviate from the steepest descent direction and still give rise to a globally convergent iteration. Various line search termination conditions can be used to establish this result, but for concreteness we will consider only the Wolfe conditions (3.6). Though Zoutendijk's result appears, at first, to be technical and obscure, its power will soon become evident.

Theorem 3.2.

Consider any iteration of the form (3.1), where p_k is a descent direction and α_k satisfies the Wolfe conditions (3.6). Suppose that f is bounded below in $\mathbb{R}^n$ and that f is continuously differentiable in an open set $\mathcal{N}$ containing the level set $\mathcal{L} \overset{\text{def}}{=} \{x : f(x) \le f(x_0)\}$, where x_0 is the starting point of the iteration. Assume also that the gradient ∇f is Lipschitz continuous on $\mathcal{N}$, that is, there exists a constant $L > 0$ such that

$$\|\nabla f(x) - \nabla f(\tilde{x})\| \le L\|x - \tilde{x}\|, \quad \text{for all } x, \tilde{x} \in \mathcal{N}. \tag{3.13}$$

Then

$$\sum_{k \ge 0} \cos^2\theta_k \, \|\nabla f_k\|^2 < \infty. \tag{3.14}$$

PROOF. From (3.6b) and (3.1) we have that

$$(\nabla f_{k+1} - \nabla f_k)^T p_k \ge (c_2 - 1)\nabla f_k^T p_k,$$

while the Lipschitz condition (3.13) implies that

$$(\nabla f_{k+1} - \nabla f_k)^T p_k \le \alpha_k L \|p_k\|^2.$$

By combining these two relations, we obtain

$$\alpha_k \ge \frac{c_2 - 1}{L} \frac{\nabla f_k^T p_k}{\|p_k\|^2}.$$

By substituting this inequality into the first Wolfe condition (3.6a), we obtain

$$f_{k+1} \le f_k - c_1 \frac{1 - c_2}{L} \frac{(\nabla f_k^T p_k)^2}{\|p_k\|^2}.$$

From the definition (3.12), we can write this relation as

$$f_{k+1} \le f_k - c \cos^2 \theta_k \|\nabla f_k\|^2,$$

where $c = c_1(1 - c_2)/L$. By summing this expression over all indices less than or equal to k, we obtain

$$f_{k+1} \le f_0 - c \sum_{j=0}^{k} \cos^2 \theta_j \|\nabla f_j\|^2. \tag{3.15}$$

Since f is bounded below, we have that $f_0 - f_{k+1}$ is less than some positive constant, for all k. Hence by taking limits in (3.15), we obtain

$$\sum_{k=0}^{\infty} \cos^2 \theta_k \|\nabla f_k\|^2 < \infty,$$

which concludes the proof. □

Similar results to this theorem hold when the Goldstein conditions (3.11) or strong Wolfe conditions (3.7) are used in place of the Wolfe conditions.

Note that the assumptions of Theorem 3.2 are not too restrictive. If the function f were not bounded below, the optimization problem would not be well-defined. The smoothness assumption—Lipschitz continuity of the gradient—is implied by many of the smoothness conditions that are used in local convergence theorems (see Chapters 6 and 8) and are often satisfied in practice.

Inequality (3.14), which we call the *Zoutendijk condition*, implies that

$$\cos^2 \theta_k \|\nabla f_k\|^2 \to 0. \tag{3.16}$$

This limit can be used in turn to derive global convergence results for line search algorithms.

If our method for choosing the search direction p_k in the iteration (3.1) ensures that the angle θ_k defined by (3.12) is bounded away from 90°, there is a positive constant δ such that

$$\cos \theta_k \ge \delta > 0, \quad \text{for all } k. \tag{3.17}$$

It follows immediately from (3.16) that

$$\lim_{k\to\infty} \|\nabla f_k\| = 0. \tag{3.18}$$

In other words, we can be sure that the gradient norms $\|\nabla f_k\|$ converge to zero, provided that the search directions are never too close to orthogonality with the gradient. In particular, the method of steepest descent (for which the search direction p_k makes an angle of *zero* degrees with the negative gradient) produces a gradient sequence that converges to zero, provided that it uses a line search satisfying the Wolfe or Goldstein conditions.

We use the term *globally convergent* to refer to algorithms for which the property (3.18) is satisfied, but note that this term is sometimes used in other contexts to mean different things. For line search methods of the general form (3.1), the limit (3.18) is the strongest global convergence result that can be obtained: We cannot guarantee that the method converges to a minimizer, but only that it is attracted by stationary points. Only by making additional requirements on the search direction p_k—by introducing negative curvature information from the Hessian $\nabla^2 f(x_k)$, for example—can we strengthen these results to include convergence to a local minimum. See the Notes and References at the end of this chapter for further discussion of this point.

Consider now the Newton-like method (3.1), (3.2) and assume that the matrices B_k are positive definite with a uniformly bounded condition number. That is, there is a constant M such that

$$\|B_k\| \, \|B_k^{-1}\| \le M, \quad \text{for all } k.$$

It is easy to show from the definition (3.12) that

$$\cos \theta_k \ge 1/M \tag{3.19}$$

(see Exercise 5). By combining this bound with (3.16) we find that

$$\lim_{k\to\infty} \|\nabla f_k\| = 0. \tag{3.20}$$

Therefore, we have shown that Newton and quasi-Newton methods are globally convergent if the matrices B_k have a bounded condition number and are positive definite (which is needed to ensure that p_k is a descent direction), and if the step lengths satisfy the Wolfe conditions.

For some algorithms, such as conjugate gradient methods, we will not be able to prove the limit (3.18), but only the weaker result

$$\liminf_{k\to\infty} \|\nabla f_k\| = 0. \tag{3.21}$$

In other words, just a subsequence of the gradient norms $\|\nabla f_{k_j}\|$ converges to zero, rather than the whole sequence (see Appendix A). This result, too, can be proved by using Zoutendijk's condition (3.14), but instead of a constructive proof, we outline a proof by contradiction. Suppose that (3.21) does not hold, so that the gradients remain bounded away from zero, that is, there exists $\gamma > 0$ such that

$$\|\nabla f_k\| \geq \gamma, \quad \text{for all } k \text{ sufficiently large.} \tag{3.22}$$

Then from (3.16) we conclude that

$$\cos\theta_k \to 0, \tag{3.23}$$

that is, the entire sequence $\{\cos\theta_k\}$ converges to 0. To establish (3.21), therefore, it is enough to show that a subsequence $\{\cos\theta_{k_j}\}$ is bounded away from zero. We will use this strategy in Chapter 5 to study the convergence of nonlinear conjugate gradient methods.

By applying this proof technique, we can prove global convergence in the sense of (3.20) or (3.21) for a general class of algorithms. Consider *any* algorithm for which (i) every iteration produces a decrease in the objective function, and (ii) every mth iteration is a steepest descent step, with step length chosen to satisfy the Wolfe or Goldstein conditions. Then since $\cos\theta_k = 1$ for the steepest descent steps, the result (3.20) holds. Of course, we would design the algorithm so that it does something "better" than steepest descent at the other $m - 1$ iterates; the occasional steepest descent steps may not make much progress, but they at least guarantee overall global convergence.

Note that throughout this section we have used only the fact that Zoutendijk's condition implies the limit (3.16). In later chapters we will make use of the bounded sum condition (3.14), which forces the sequence $\{\cos^2\theta_k \|\nabla f_k\|^2\}$ to converge to zero at a sufficiently rapid rate.

3.3 RATE OF CONVERGENCE

It would seem that designing optimization algorithms with good convergence properties is easy, since all we need to ensure is that the search direction p_k does not tend to become orthogonal to the gradient ∇f_k, or that steepest descent steps are taken regularly. We could

simply compute $\cos\theta_k$ at every iteration and turn p_k toward the steepest descent direction if $\cos\theta_k$ is smaller than some preselected constant $\delta > 0$. Angle tests of this type ensure global convergence, but they are undesirable for two reasons. First, they may impede a fast rate of convergence, because for problems with an ill-conditioned Hessian, it may be necessary to produce search directions that are almost orthogonal to the gradient, and an inappropriate choice of the parameter δ may prevent this. Second, angle tests destroy the invariance properties of quasi-Newton methods.

Algorithmic strategies that achieve rapid convergence can sometimes conflict with the requirements of global convergence, and vice versa. For example, the steepest descent method is the quintessential globally convergent algorithm, but it is quite slow in practice, as we shall see below. On the other hand, the pure Newton iteration converges rapidly when started close enough to a solution, but its steps may not even be descent directions away from the solution. The challenge is to design algorithms that incorporate both properties: good global convergence guarantees and a rapid rate of convergence.

We begin our study of convergence rates of line search methods by considering the most basic approach of all: the steepest descent method.

CONVERGENCE RATE OF STEEPEST DESCENT

We can learn much about the steepest descent method by considering the ideal case, in which the objective function is quadratic and the line searches are exact. Let us suppose that

$$f(x) = \tfrac{1}{2}x^T Q x - b^T x, \tag{3.24}$$

where Q is symmetric and positive definite. The gradient is given by $\nabla f(x) = Qx - b$, and the minimizer x^* is the unique solution of the linear system $Qx = b$.

Let us compute the step length α_k that minimizes $f(x_k - \alpha\nabla f_k)$. By differentiating

$$f(x_k - \alpha g_k) = \frac{1}{2}(x_k - \alpha g_k)^T Q(x_k - \alpha g_k) - b^T(x_k - \alpha g_k)$$

with respect to α, we obtain

$$\alpha_k = \frac{\nabla f_k^T \nabla f_k}{\nabla f_k^T Q \nabla f_k}. \tag{3.25}$$

If we use this exact minimizer α_k, the steepest descent iteration for (3.24) is given by

$$x_{k+1} = x_k - \left(\frac{\nabla f_k^T \nabla f_k}{\nabla f_k^T Q \nabla f_k}\right)\nabla f_k. \tag{3.26}$$

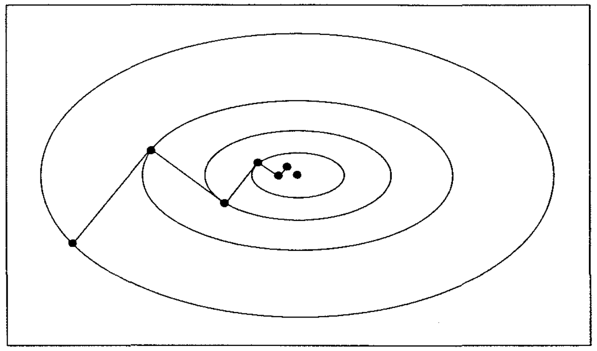


Figure 3.7 Steepest descent steps.

Since $\nabla f_k = Qx_k - b$, this equation yields a closed-form expression for x_{k+1} in terms of x_k. In Figure 3.7 we plot a typical sequence of iterates generated by the steepest descent method on a two-dimensional quadratic objective function. The contours of f are ellipsoids whose axes lie along the orthogonal eigenvectors of Q. Note that the iterates zigzag toward the solution.

To quantify the rate of convergence we introduce the weighted norm $\|x\|_Q^2 = x^T Qx$. By using the relation $Qx^* = b$, we can show easily that

$$\tfrac{1}{2}\|x - x^*\|_Q^2 = f(x) - f(x^*), \tag{3.27}$$

so that this norm measures the difference between the current objective value and the optimal value. By using the equality (3.26) and noting that $\nabla f_k = Q(x_k - x^*)$, we can derive the equality

$$\|x_{k+1} - x^*\|_Q^2 = \left\{1 - \frac{(\nabla f_k^T \nabla f_k)^2}{(\nabla f_k^T Q \nabla f_k)(\nabla f_k^T Q^{-1} \nabla f_k)}\right\} \|x_k - x^*\|_Q^2 \tag{3.28}$$

(see Exercise 7). This expression describes the exact decrease in f at each iteration, but since the term inside the brackets is difficult to interpret, it is more useful to bound it in terms of the condition number of the problem.

Theorem 3.3.

When the steepest descent method with exact line searches (3.26) is applied to the strongly convex quadratic function (3.24), the error norm (3.27) satisfies

$$\|x_{k+1} - x^*\|_Q^2 \le \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)^2 \|x_k - x^*\|_Q^2, \tag{3.29}$$

where $0 < \lambda_1 \le \cdots \le \lambda_n$ *are the eigenvalues of* Q.

The proof of this result is given by Luenberger [152]. The inequalities (3.29) and (3.27) show that the function values f_k converge to the minimum f_* at a linear rate. As a special case of this result, we see that convergence is achieved in one iteration if all the eigenvalues are equal. In this case, Q is a multiple of the identity matrix, so the contours in Figure 3.7 are circles and the steepest descent direction always points at the solution. In general, as the condition number $\kappa(Q) = \lambda_n/\lambda_1$ increases, the contours of the quadratic become more elongated, the zigzagging in Figure 3.7 becomes more pronounced, and (3.29) implies that the convergence degrades. Even though (3.29) is a worst-case bound, it gives an accurate indication of the behavior of the algorithm when $n > 2$.

The rate-of-convergence behavior of the steepest descent method is essentially the same on general nonlinear objective functions. In the following result we assume that the step length is the global minimizer along the search direction.

Theorem 3.4.

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ *is twice continuously differentiable, and that the iterates generated by the steepest descent method with exact line searches converge to a point* x^* *where the Hessian matrix* $\nabla^2 f(x^*)$ *is positive definite. Then*

$$f(x_{k+1}) - f(x^*) \le \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)^2 [f(x_k) - f(x^*)],$$

where $\lambda_1 \le \cdots \le \lambda_n$ *are the eigenvalues of* $\nabla^2 f(x^*)$.

In general, we cannot expect the rate of convergence to improve if an inexact line search is used. Therefore, Theorem 3.4 shows that the steepest descent method can have an unacceptably slow rate of convergence, even when the Hessian is reasonably well conditioned. For example, if $\kappa(Q) = 800$, $f(x_1) = 1$ and $f(x^*) = 0$, Theorem 3.4 suggests that the function value will still be about 0.08 after one thousand iterations of the steepest descent method.

QUASI-NEWTON METHODS

Let us now suppose that the search direction has the form

$$p_k = -B_k^{-1} \nabla f_k, \tag{3.30}$$

where the symmetric and positive definite matrix B_k is updated at every iteration by a quasi-Newton updating formula. We already encountered one quasi-Newton formula, the BFGS formula, in Chapter 2; others will be discussed in Chapter 8. We assume here that the step length α_k will be computed by an inexact line search that satisfies the Wolfe or strong Wolfe conditions, with one important proviso: The line search algorithm will always try the step length $\alpha = 1$ first, and will accept this value if it satisfies the Wolfe conditions. (We could enforce this condition by setting $\bar{\alpha} = 1$ in Procedure 3.1, for example.) This implementation detail turns out to be crucial in obtaining a fast rate of convergence.

The following result, due to Dennis and Moré, shows that if the search direction of a quasi-Newton method approximates the Newton direction well enough, then the unit step length will satisfy the Wolfe conditions as the iterates converge to the solution. It also specifies a condition that the search direction must satisfy in order to give rise to a superlinearly convergent iteration. To bring out the full generality of this result, we state it first in terms of a general descent iteration, and then examine its consequences for quasi-Newton and Newton methods.

Theorem 3.5.

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is three times continuously differentiable. Consider the iteration $x_{k+1} = x_k + \alpha_k p_k$, where p_k is a descent direction and α_k satisfies the Wolfe conditions (3.6) with $c_1 \le \frac{1}{2}$. If the sequence $\{x_k\}$ converges to a point x^* such that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite, and if the search direction satisfies

$$\lim_{k\to\infty} \frac{\|\nabla f_k + \nabla^2 f_k p_k\|}{\|p_k\|} = 0, \tag{3.31}$$

then

(i) *the step length $\alpha_k = 1$ is admissible for all k greater than a certain index k_0; and*

(ii) *if $\alpha_k = 1$ for all $k > k_0$, $\{x_k\}$ converges to x^* superlinearly.*

It is easy to see that if $c_1 > \frac{1}{2}$, then the line search would exclude the minimizer of a quadratic, and unit step lengths may not be admissible.

If p_k is a quasi-Newton search direction of the form (3.30), then (3.31) is equivalent to

$$\lim_{k\to\infty} \frac{\|(B_k - \nabla^2 f(x^*))p_k\|}{\|p_k\|} = 0. \tag{3.32}$$

Hence, we have the surprising (and delightful) result that a superlinear convergence rate can be attained even if the sequence of quasi-Newton matrices B_k does not converge to $\nabla^2 f(x^*)$; it suffices that the B_k become increasingly accurate approximations to $\nabla^2 f(x^*)$ *along the search directions p_k*.

An important remark is that condition (3.32) is both necessary and sufficient for the superlinear convergence of quasi-Newton methods.

Theorem 3.6.

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is three times continuously differentiable. Consider the iteration $x_{k+1} = x_k + p_k$ (that is, the step length α_k is uniformly 1) and that p_k is given by (3.30). Let us also assume that $\{x_k\}$ converges to a point x^* such that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite. Then $\{x_k\}$ converges superlinearly if and only if (3.32) holds.

PROOF. We first show that (3.32) is equivalent to

$$p_k - p_k^{\mathrm{N}} = o(\|p_k\|), \tag{3.33}$$

where $p_k^{\mathrm{N}} = -\nabla^2 f_k^{-1} \nabla f_k$ is the Newton step. Assuming that (3.32) holds, we have that

$$\begin{aligned} p_k - p_k^{\mathrm{N}} &= \nabla^2 f_k^{-1}(\nabla^2 f_k p_k + \nabla f_k) \\ &= \nabla^2 f_k^{-1}(\nabla^2 f_k - B_k) p_k \\ &= O(\|(\nabla^2 f_k - B_k) p_k\|) \\ &= o(\|p_k\|), \end{aligned}$$

where we have used the fact that $\|\nabla^2 f_k^{-1}\|$ is bounded above for x_k sufficiently close to x^*, since the limiting Hessian $\nabla^2 f(x^*)$ is positive definite. The converse follows readily if we multiply both sides of (3.33) by $\nabla^2 f_k$ and recall (3.30).

For the remainder of the proof we need to look ahead to the proof of quadratic convergence of Newton's method and, in particular, the estimate (3.37). By using this inequality together with (3.33), we obtain that

$$\begin{aligned} \|x_k + p_k - x^*\| &\le \|x_k + p_k^{\mathrm{N}} - x^*\| + \|p_k - p_k^{\mathrm{N}}\| \\ &= O(\|x_k - x^*\|^2) + o(\|p_k\|). \end{aligned}$$

A simple manipulation of this inequality reveals that $\|p_k\| = O(\|x_k - x^*\|)$, so we obtain

$$\|x_k + p_k - x^*\| \le o(\|x_k - x^*\|),$$

giving the superlinear convergence result. □

We will see in Chapter 8 that quasi-Newton methods normally satisfy condition (3.32) and are superlinearly convergent.

NEWTON'S METHOD

Let us now consider the Newton iteration where the search direction is given by

$$p_k^{\mathrm{N}} = -\nabla^2 f_k^{-1} \nabla f_k. \tag{3.34}$$

Since the Hessian matrix $\nabla^2 f_k$ may not always be positive definite, p_k^{N} may not always be a descent direction, and many of the ideas discussed so far in this chapter no longer apply. In Chapter 6 we will describe two approaches for obtaining a globally convergent iteration based on the Newton step: a line search approach, in which the Hessian $\nabla^2 f_k$ is modified, if necessary, to make it positive definite and thereby yield descent, and a trust region approach, in which $\nabla^2 f_k$ is used to form a quadratic model that is minimized in a ball.

Here we discuss just the local rate-of-convergence properties of Newton's method. We know that for all x in the vicinity of a solution point x^* such that $\nabla^2 f(x^*)$ is positive definite, the Hessian $\nabla^2 f(x)$ will also be positive definite. Newton's method will be well-defined in this region and will converge quadratically, provided that the step lengths α_k are eventually always 1.

Theorem 3.7.

Suppose that f is twice differentiable and that the Hessian $\nabla^2 f(x)$ is Lipschitz continuous (see (A.8)) in a neighborhood of a solution x^* at which the sufficient conditions (Theorem 2.4) are satisfied. Consider the iteration $x_{k+1} = x_k + p_k$, where p_k is given by (3.34). Then

1. *if the starting point x_0 is sufficiently close to x^*, the sequence of iterates converges to x^*;*
2. *the rate of convergence of $\{x_k\}$ is quadratic; and*
3. *the sequence of gradient norms $\{\|\nabla f_k\|\}$ converges quadratically to zero.*

PROOF. From the definition of the Newton step and the optimality condition $\nabla f_* = 0$ we have that

$$\begin{aligned} x_k + p_k^{\mathrm{N}} - x^* &= x_k - x^* - \nabla^2 f_k^{-1} \nabla f_k \\ &= \nabla^2 f_k^{-1} \left[\nabla^2 f_k (x_k - x^*) - (\nabla f_k - \nabla f_*) \right]. \end{aligned} \tag{3.35}$$

Since

$$\nabla f_k - \nabla f_* = \int_0^1 \nabla^2 f(x_k + t(x^* - x_k))(x_k - x^*)\, dt,$$

we have

$$\begin{aligned} &\left\| \nabla^2 f(x_k)(x_k - x^*) - (\nabla f_k - \nabla f(x^*)) \right\| \\ &\quad = \left\| \int_0^1 \left[\nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^* - x_k)) \right] (x_k - x^*)\, dt \right\| \\ &\quad \le \int_0^1 \left\| \nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^* - x_k)) \right\| \|x_k - x^*\|\, dt \\ &\quad \le \|x_k - x^*\|^2 \int_0^1 Lt\, dt = \tfrac{1}{2} L \|x_k - x^*\|^2, \end{aligned} \tag{3.36}$$

where L is the Lipschitz constant for $\nabla^2 f(x)$ for x near x^*. Since $\nabla^2 f(x^*)$ is nonsingular, and since $\nabla^2 f_k \to \nabla^2 f(x^*)$, we have that $\|\nabla^2 f_k^{-1}\| \le 2\|\nabla^2 f(x^*)^{-1}\|$ for all k sufficiently large. By substituting in (3.35) and (3.36), we obtain

$$\|x_k + p_k^{\mathrm{N}} - x^*\| \le L \|\nabla^2 f(x^*)^{-1}\| \|x_k - x^*\|^2 = \tilde{L} \|x_k - x^*\|^2, \tag{3.37}$$

where $\tilde{L} = L\|\nabla^2 f(x^*)^{-1}\|$. Using this inequality inductively we deduce that if the starting point is sufficiently near x^*, then the sequence converges to x^*, and the rate of convergence is quadratic.

By using the relations $x_{k+1} - x_k = p_k^{\mathrm{N}}$ and $\nabla f_k + \nabla^2 f_k p_k^{\mathrm{N}} = 0$, we obtain that

$$\begin{aligned} \|\nabla f(x_{k+1})\| &= \|\nabla f(x_{k+1}) - \nabla f_k - \nabla^2 f(x_k) p_k^{\mathrm{N}}\| \\ &= \left\| \int_0^1 \nabla^2 f(x_k + t p_k^{\mathrm{N}})(x_{k+1} - x_k)\, dt - \nabla^2 f(x_k) p_k^{\mathrm{N}} \right\| \\ &\le \int_0^1 \left\| \nabla^2 f(x_k + t p_k^{\mathrm{N}}) - \nabla^2 f(x_k) \right\| \|p_k^{\mathrm{N}}\|\, dt \\ &\le \tfrac{1}{2} L \|p_k^{\mathrm{N}}\|^2 \\ &\le \tfrac{1}{2} L \|\nabla^2 f(x_k)^{-1}\|^2 \|\nabla f_k\|^2 \\ &\le 2L \|\nabla^2 f(x^*)^{-1}\|^2 \|\nabla f_k\|^2, \end{aligned}$$

proving that the gradient norms converge to zero quadratically. □

When the search direction is given by Newton's method, the limit (3.31) is satisfied (the ratio is zero for all k!), and Theorem 3.5 shows that the Wolfe conditions will accept the step length α_k for all large k. The same is true of the Goldstein conditions. Thus implementations of Newton's method using these conditions, and in which the line search always tries the unit step length first, will set $\alpha_k = 1$ for all large k and attain a local quadratic rate of convergence.

COORDINATE DESCENT METHODS

An approach that is frequently used in practice is to cycle through the n coordinate directions $e_1, e_2, \ldots, e_n$, using each in turn as a search direction. At the first iteration, we fix all except the first variable, and find a new value of this variable that minimizes (or at least reduces) the objective function. On the next iteration, we repeat the process with the *second* variable, and so on. After n iterations, we return to the first variable and repeat the cycle. The method is referred to as the *method of alternating variables* or the *coordinate descent method.* Though simple and somewhat intuitive, it can be quite inefficient in practice, as we illustrate in Figure 3.8 for a quadratic function in two variables. Note that after a few iterations, neither the vertical nor the horizontal move makes much progress toward the solution.

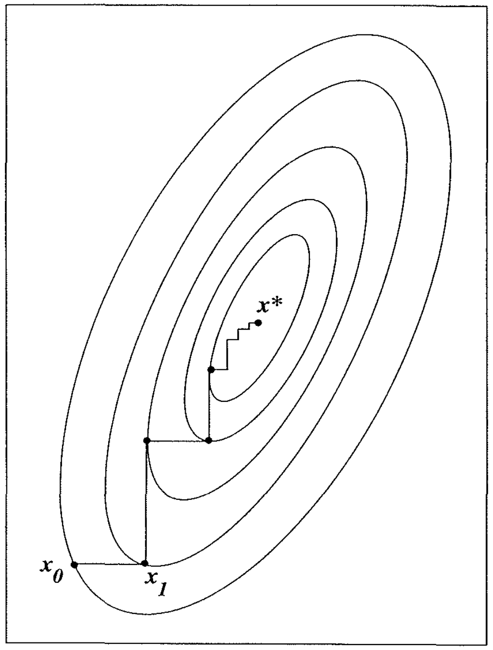


Figure 3.8 Coordinate descent.

In fact, the coordinate descent method with exact line searches can iterate infinitely without ever approaching a point where the gradient of the objective function vanishes. (By contrast, the steepest descent method produces a sequence for which $\|\nabla f_k\| \to 0$, as we showed earlier.) This observation can be generalized to show that a cyclic search along *any* set of linearly independent directions does not guarantee global convergence (Powell [198]). The difficulty that arises is that the gradient ∇f_k may become more and more perpendicular to the coordinate search direction, so that $\cos\theta_k$ approaches zero sufficiently rapidly that the Zoutendijk condition (3.14) is satisfied even when ∇f_k does not approach zero.

If the coordinate descent method converges to a solution, then its rate of convergence is often much slower than that of the steepest descent method, and the difference between them increases with the number of variables. However, the method may still be useful because

it does not require calculation of the gradient ∇f_k, and

the speed of convergence can be quite acceptable if the variables are loosely coupled.

Many variants of the coordinate descent method have been proposed, some of which are globally convergent. One simple variant is a "back-and-forth" approach in which we

search along the sequence of directions

$$e_1, e_2, \ldots, e_{n-1}, e_n, e_{n-1}, \ldots, e_2, e_1, e_2, \ldots \quad \text{(repeats)}.$$

Another approach, suggested by Figure 3.8, is first to perform a sequence of coordinate descent steps and then search along the line joining the first and last points in the cycle. Several algorithms, such as that of Hooke and Jeeves, are based on these ideas; see [104, 83].

3.4 STEP-LENGTH SELECTION ALGORITHMS

We now consider techniques for finding a minimum of the one-dimensional function

$$\phi(\alpha) = f(x_k + \alpha p_k), \tag{3.38}$$

or for simply finding a step length α_k satisfying one of the termination conditions described in Section 3.1. We assume that p_k is a descent direction—that is, $\phi'(0) < 0$—so that our search can be confined to positive values of α.

If f is a convex quadratic, $f(x) = \frac{1}{2}x^T Qx + b^T x + c$, its one-dimensional minimizer along the ray $x_k + \alpha p_k$ can be computed analytically and is given by

$$\alpha_k = -\frac{\nabla f_k^T p_k}{p_k^T Q p_k}. \tag{3.39}$$

For general nonlinear functions, it is necessary to use an iterative procedure. Much attention must be given to this line search because it has a major impact on the robustness and efficiency of all nonlinear optimization methods.

Line search procedures can be classified according to the type of derivative information they use. Algorithms that use only function values can be inefficient, since to be theoretically sound, they need to continue iterating until the search for the minimizer is narrowed down to a small interval. In contrast, knowledge of gradient information allows us to determine whether a suitable step length has been located, as stipulated, for example, by the Wolfe conditions (3.6) or Goldstein conditions (3.11). Often, particularly when the iterates are close to the solution, the very first step satisfies these conditions, so the line search need not be invoked at all. In the rest of this section we will discuss only algorithms that make use of derivative information. More information on derivative-free procedures is given in the notes at the end of this chapter.

All line search procedures require an initial estimate α_0 and generate a sequence $\{\alpha_i\}$ that either terminates with a step length satisfying the conditions specified by the user (for example, the Wolfe conditions) or determines that such a step length does not exist. Typical procedures consist of two phases: a *bracketing phase* that finds an interval $[a, b]$ containing acceptable step lengths, and a *selection phase* that zooms in to locate the final step length. The selection phase usually reduces the bracketing interval during its search for the desired step length and interpolates some of the function and derivative information gathered on earlier steps to guess the location of the minimizer. We will first discuss how to perform this interpolation.

In the following discussion we let α_k and α_{k-1} denote the step lengths used at iterations k and $k-1$ of the optimization algorithm, respectively. On the other hand, we denote the trial step lengths generated during the line search by α_i and α_{i-1} and also α_j. We use α_0 to denote the initial guess.

INTERPOLATION

We begin by describing a line search procedure based on interpolation of known function and derivative values of the function ϕ. This procedure can be viewed as an enhancement of Procedure 3.1. The aim is to find a value of α that satisfies the sufficient decrease condition (3.6a), without being "too small." Accordingly, the procedures here generate a decreasing sequence of values α_i such that each value α_i is not too much smaller than its predecessor α_{i-1}.

Note that we can write the sufficient decrease condition in the notation of (3.38) as

$$\phi(\alpha_k) \le \phi(0) + c_1\alpha_k\phi'(0), \tag{3.40}$$

and that since the constant c_1 is usually chosen to be small in practice ($c_1 = 10^{-4}$, say), this condition asks for little more than descent in f. We design the procedure to be "efficient" in the sense that it computes the derivative $\nabla f(x)$ as few times as possible.

Suppose that the initial guess α_0 is given. If we have

$$\phi(\alpha_0) \le \phi(0) + c_1\alpha_0\phi'(0),$$

this step length satisfies the condition, and we terminate the search. Otherwise, we know that the interval $[0, \alpha_0]$ contains acceptable step lengths (see Figure 3.3). We form a quadratic approximation $\phi_q(\alpha)$ to ϕ by interpolating the three pieces of information available—$\phi(0)$, $\phi'(0)$, and $\phi(\alpha_0)$—to obtain

$$\phi_q(\alpha) = \left(\frac{\phi(\alpha_0) - \phi(0) - \alpha_0\phi'(0)}{\alpha_0^2}\right)\alpha^2 + \phi'(0)\alpha + \phi(0). \tag{3.41}$$

(Note that this function is constructed so that it satisfies the interpolation conditions $\phi_q(0) = \phi(0)$, $\phi_q'(0) = \phi'(0)$, and $\phi_q(\alpha_0) = \phi(\alpha_0)$.) The new trial value α_1 is defined as the minimizer of this quadratic, that is, we obtain

$$\alpha_1 = -\frac{\phi'(0)\alpha_0^2}{2\left[\phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0\right]}. \tag{3.42}$$

If the sufficient decrease condition (3.40) is satisfied at α_1, we terminate the search. Otherwise, we construct a *cubic* function that interpolates the four pieces of information $\phi(0)$, $\phi'(0)$, $\phi(\alpha_0)$, and $\phi(\alpha_1)$, obtaining

$$\phi_c(\alpha) = a\alpha^3 + b\alpha^2 + \alpha\phi'(0) + \phi(0),$$

where

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\alpha_0^2\alpha_1^2(\alpha_1 - \alpha_0)} \begin{bmatrix} \alpha_0^2 & -\alpha_1^2 \\ -\alpha_0^3 & \alpha_1^3 \end{bmatrix} \begin{bmatrix} \phi(\alpha_1) - \phi(0) - \phi'(0)\alpha_1 \\ \phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0 \end{bmatrix}.$$

By differentiating $\phi_c(x)$, we see that the minimizer α_2 of ϕ_c lies in the interval $[0, \alpha_1]$ and is given by

$$\alpha_2 = \frac{-b + \sqrt{b^2 - 3a\phi'(0)}}{3a}.$$

If necessary, this process is repeated, using a cubic interpolant of $\phi(0)$, $\phi'(0)$ and the two most recent values of ϕ, until an α that satisfies (3.40) is located. If any α_i is either too close to its predecessor α_{i-1} or else too much smaller than α_{i-1}, we reset $\alpha_i = \alpha_{i-1}/2$. This safeguard procedure ensures that we make reasonable progress on each iteration and that the final α is not too small.

The strategy just described assumes that derivative values are significantly more expensive to compute than function values. It is often possible, however, to compute the directional derivative simultaneously with the function, at little additional cost; see Chapter 7. Accordingly, we can design an alternative strategy based on cubic interpolation of the values of ϕ and ϕ' at the two most recent values of α.

Cubic interpolation provides a good model for functions with significant changes of curvature. Suppose we have an interval $[a, b]$ known to contain desirable step lengths, and two previous step length estimates α_{i-1} and α_i in this interval. We use a cubic function to interpolate $\phi(\alpha_{i-1})$, $\phi'(\alpha_{i-1})$, $\phi(\alpha_i)$, and $\phi'(\alpha_i)$. (This cubic function always exists and is unique; see, for example, Bulirsch and Stoer [29, p. 52].) The minimizer of this cubic in $[a, b]$ is either at one of the endpoints or else in the interior, in which case it is given by

$$\alpha_{i+1} = \alpha_i - (\alpha_i - \alpha_{i-1})\left[\frac{\phi'(\alpha_i) + d_2 - d_1}{\phi'(\alpha_i) - \phi'(\alpha_{i-1}) + 2d_2}\right], \tag{3.43}$$

with

$$d_1 = \phi'(\alpha_{i-1}) + \phi'(\alpha_i) - 3\frac{\phi(\alpha_{i-1}) - \phi(\alpha_i)}{\alpha_{i-1} - \alpha_i},$$
$$d_2 = \left[d_1^2 - \phi'(\alpha_{i-1})\phi'(\alpha_i)\right]^{1/2}.$$

The interpolation process can be repeated by discarding the data at one of the step lengths α_{i-1} or α_i and replacing it by $\phi(\alpha_{i+1})$ and $\phi'(\alpha_{i+1})$. The decision on which of α_{i-1} and α_i should be kept and which discarded depends on the specific conditions used to terminate the line search; we discuss this issue further below in the context of the Wolfe conditions. Cubic interpolation is a powerful strategy, since it can produce a quadratic rate of convergence of the iteration (3.43) to the minimizing value of α.

THE INITIAL STEP LENGTH

For Newton and quasi-Newton methods the step $\alpha_0 = 1$ should always be used as the initial trial step length. This choice ensures that unit step lengths are taken whenever they satisfy the termination conditions and allows the rapid rate-of-convergence properties of these methods to take effect.

For methods that do not produce well-scaled search directions, such as the steepest descent and conjugate gradient methods, it is important to use current information about the problem and the algorithm to make the initial guess. A popular strategy is to assume that the first-order change in the function at iterate x_k will be the same as that obtained at the previous step. In other words, we choose the initial guess α_0 so that $\alpha_0 \nabla f_k^T p_k = \alpha_{k-1} \nabla f_{k-1}^T p_{k-1}$. We therefore have

$$\alpha_0 = \alpha_{k-1} \frac{\nabla f_{k-1}^T p_{k-1}}{\nabla f_k^T p_k}.$$

Another useful strategy is to interpolate a quadratic to the data $f(x_{k-1})$, $f(x_k)$, and $\phi'(0) = \nabla f_k^T p_k$ and to define α_0 to be its minimizer. This strategy yields

$$\alpha_0 = \frac{2(f_k - f_{k-1})}{\phi'(0)}. \tag{3.44}$$

It can be shown that if $x_k \to x^*$ superlinearly, then the ratio in this expression converges to 1. If we adjust the choice (3.44) by setting

$$\alpha_0 \leftarrow \min(1, 1.01\alpha_0),$$

we find that the unit step length $\alpha_0 = 1$ will eventually always be tried and accepted, and the superlinear convergence properties of Newton and quasi-Newton methods will be observed.

A LINE SEARCH ALGORITHM FOR THE WOLFE CONDITIONS

The Wolfe (or strong Wolfe) conditions are among the most widely applicable and useful termination conditions. We now describe in some detail a one-dimensional search procedure that is guaranteed to find a step length satisfying the *strong* Wolfe conditions (3.7) for any parameters c_1 and c_2 satisfying $0 < c_1 < c_2 < 1$. As before, we assume that p is a descent direction and that f is bounded below along the direction p.

The algorithm has two stages. This first stage begins with a trial estimate α_1, and keeps increasing it until it finds either an acceptable step length or an interval that brackets the desired step lengths. In the latter case, the second stage is invoked by calling a function called **zoom** (Algorithm 3.3 below), which successively decreases the size of the interval until an acceptable step length is identified.

A formal specification of the line search algorithm follows. We refer to (3.7a) as the *sufficient decrease condition* and to (3.7b) as the *curvature condition*. The parameter $\alpha_{\max}$ is a user-supplied bound on the maximum step length allowed. The line search algorithm terminates with α_* set to a step length that satisfies the strong Wolfe conditions.

Algorithm 3.2 (Line Search Algorithm).

```
Set $\alpha_0 \leftarrow 0$, choose $\alpha_1 > 0$ and $\alpha_{\max}$;
$i \leftarrow 1$;
repeat
    Evaluate $\phi(\alpha_i)$;
    if $\phi(\alpha_i) > \phi(0) + c_1\alpha_i\phi'(0)$ or $[\phi(\alpha_i) \geq \phi(\alpha_{i-1})$ and $i > 1]$
        $\alpha_* \leftarrow$zoom$(\alpha_{i-1}, \alpha_i)$ and stop;
    Evaluate $\phi'(\alpha_i)$;
    if $|\phi'(\alpha_i)| \leq -c_2\phi'(0)$
        set $\alpha_* \leftarrow \alpha_i$ and stop;
    if $\phi'(\alpha_i) \geq 0$
        set $\alpha_* \leftarrow$zoom$(\alpha_i, \alpha_{i-1})$ and stop;
    Choose $\alpha_{i+1} \in (\alpha_i, \alpha_{\max})$
    $i \leftarrow i + 1$;
end (repeat)
```

Note that the sequence of trial step lengths $\{\alpha_i\}$ is monotonically increasing, but that the order of the arguments supplied to the **zoom** function may vary. The procedure uses the knowledge that the interval (α_{i-1}, α_i) contains step lengths satisfying the strong Wolfe conditions if one of the following three conditions is satisfied:

(i) α_i violates the sufficient decrease condition;

(ii) $\phi(\alpha_i) \geq \phi(\alpha_{i-1})$;

(iii) $\phi'(\alpha_i) \geq 0$.

The last step of the algorithm performs extrapolation to find the next trial value α_{i+1}. To implement this step we can use approaches like the interpolation procedures above, or we can simply set α_{i+1} to some constant multiple of α_i. Whichever strategy we use, it is important that the successive steps increase quickly enough to reach the upper limit $\alpha_{\max}$ in a finite number of iterations.

We now specify the function **zoom**, which requires a little explanation. The order of its input arguments is such that each call has the form **zoom**($\alpha_{\text{lo}}, \alpha_{\text{hi}}$), where

(a) the interval bounded by α_{lo} and α_{hi} contains step lengths that satisfy the strong Wolfe conditions;

(b) α_{lo} is, among all step lengths generated so far and satisfying the sufficient decrease condition, the one giving the smallest function value; and

(c) α_{hi} is chosen so that $\phi'(\alpha_{\text{lo}})(\alpha_{\text{hi}} - \alpha_{\text{lo}}) < 0$.

Each iteration of **zoom** generates an iterate α_j between α_{lo} and α_{hi}, and then replaces one of these endpoints by α_j in such a way that the properties (a), (b), and (c) continue to hold.

Algorithm 3.3 (zoom).

```
repeat
    Interpolate (using quadratic, cubic, or bisection) to find
        a trial step length $\alpha_j$ between $\alpha_{\text{lo}}$ and $\alpha_{\text{hi}}$;
    Evaluate $\phi(\alpha_j)$;
    if $\phi(\alpha_j) > \phi(0) + c_1\alpha_j\phi'(0)$ or $\phi(\alpha_j) \geq \phi(\alpha_{\text{lo}})$
        $\alpha_{\text{hi}} \leftarrow \alpha_j$;
    else
        Evaluate $\phi'(\alpha_j)$;
        if $|\phi'(\alpha_j)| \leq -c_2\phi'(0)$
            Set $\alpha_* \leftarrow \alpha_j$ and stop;
        if $\phi'(\alpha_j)(\alpha_{\text{hi}} - \alpha_{\text{lo}}) \geq 0$
            $\alpha_{\text{hi}} \leftarrow \alpha_{\text{lo}}$;
        $\alpha_{\text{lo}} \leftarrow \alpha_j$;
end (repeat)
```

If the new estimate α_j happens to satisfy the strong Wolfe conditions, then **zoom** has served its purpose of identifying such a point, so it terminates with $\alpha_* = \alpha_j$. Otherwise, if α_j satisfies the sufficient decrease condition and has a lower function value than x_{lo}, then we set $\alpha_{\text{lo}} \leftarrow \alpha_j$ to maintain condition (b). If this results in a violation of condition (c), we remedy the situation by setting α_{hi} to the old value of α_{lo}. The reader should sketch some graphs to illustrate the workings of **zoom**!

As mentioned earlier, the interpolation step that determines α_j should be safeguarded to ensure that the new step length is not too close to the endpoints of the interval. Practical line search algorithms also make use of the properties of the interpolating polynomials to make educated guesses of where the next step length should lie; see [27, 172]. A problem that can arise in the implementation is that as the optimization algorithm approaches the

solution, two consecutive function values $f(x_k)$ and $f(x_{k-1})$ may be indistinguishable in finite-precision arithmetic. Therefore, the line search must include a stopping test if it cannot attain a lower function value after a certain number (typically, ten) of trial step lengths. Some procedures also stop if the relative change in x is close to machine accuracy, or to some user-specified threshold.

A line search algorithm that incorporates all these features is difficult to code. We advocate the use of one of the several good software implementations available in the public domain. See Dennis and Schnabel [69], Lemaréchal [149], Fletcher [83], and in particular Moré and Thuente [172].

One may ask how much more expensive it is to require the strong Wolfe conditions instead of the regular Wolfe conditions. Our experience suggests that for a "loose" line search (with parameters such as $c_1 = 10^{-4}$ and $c_2 = 0.9$), both strategies require a similar amount of work. The strong Wolfe conditions have the advantage that by decreasing c_2 we can directly control the quality of the search by forcing the accepted value of α to lie closer to a local minimum. This feature is important in steepest descent or nonlinear conjugate gradient methods, and therefore a step selection routine that enforces the strong Wolfe conditions is of wider applicability.

NOTES AND REFERENCES

For an extensive discussion of line search termination conditions see Ortega and Rheinboldt [185]. Akaike [2] presents a probabilistic analysis of the steepest descent method with exact line searches on quadratic functions. He shows that when $n > 2$, the worst-case bound (3.28) can be expected to hold for most starting points. The case where $n = 2$ can be studied in closed form; see Bazaraa, Sherali, and Shetty [7].

Some line search methods (see Goldfarb [113] and Moré and Sorensen [169]) compute a direction of negative curvature, whenever it exists, to prevent the iteration from converging to nonminimizing stationary points. A direction of negative curvature p_- is one that satisfies $p_-^T \nabla^2 f(x_k) p_- < 0$. These algorithms generate a search direction by combining p_- with the steepest descent direction $-\nabla f$, and often perform a curvilinear backtracking line search. It is difficult to determine the relative contributions of the steepest descent and negative curvature directions, and due to this, this approach fell out of favor after the introduction of trust-region methods.

For a discussion on the rate of convergence of the coordinate descent method and for more references about this method see Luenberger [152].

Derivative-free line search algorithms include golden section and Fibonacci search. They share some of the features with the line search method given in this chapter. They typically store three trial points that determine an interval containing a one-dimensional minimizer. Golden section and Fibonacci differ in the way in which the trial step lengths are generated; see, for example, [58, 27].

Our discussion of interpolation follows Dennis and Schnabel [69], and the algorithm for finding a step length satisfying the strong Wolfe conditions can be found in Fletcher [83].

EXERCISES

3.1 Program the steepest descent and Newton algorithms using the backtracking line search, Procedure 3.1. Use them to minimize the Rosenbrock function (2.23). Set the initial step length $\alpha_0 = 1$ and print the step length used by each method at each iteration. First try the initial point $x_0 = (1.2, 1.2)$ and then the more difficult point $x_0 = (-1.2, 1)$.

3.2 Show that if $0 < c_2 < c_1 < 1$, then there may be no step lengths that satisfy the Wolfe conditions.

3.3 Show that the one-dimensional minimizer of a strongly convex quadratic function is given by (3.39).

3.4 Show that if $c \le \frac{1}{2}$, then the one-dimensional minimizer of a strongly convex quadratic function always satisfies the Goldstein conditions (3.11).

3.5 Prove that $\|Bx\| \ge \|x\|/\|B^{-1}\|$ for any nonsingular matrix B. Use this to establish (3.19).

3.6 Consider the steepest descent method with exact line searches applied to the convex quadratic function (3.24). Using the properties given in this chapter, show that if the initial point is such that $x_0 - x^*$ is parallel to an eigenvector of Q, then the steepest descent method will find the solution in one step.

3.7 Prove the result (3.28) by working through the following steps. First, use (3.26) to show that

$$\|x_k - x^*\|_Q^2 - \|x_{k+1} - x^*\|_Q^2 = 2\alpha_k \nabla f_k^T Q(x_k - x^*) - \alpha_k^2 \nabla f_k^T Q \nabla f_k,$$

where $\|\cdot\|_Q$ is defined by (3.27). Second, use the fact that $\nabla f_k = Q(x_k - x^*)$ to obtain

$$\|x_k - x^*\|_Q^2 - \|x_{k+1} - x^*\|_Q^2 = \frac{2(g_k^T g_k)^2}{(g_k^T Q g_k)} - \frac{(g_k^T g_k)^2}{(g_k^T Q g_k)}$$

and

$$\|x_k - x^*\|_Q^2 = \nabla f_k^T Q^{-1} \nabla f_k.$$

3.8 Let Q be a positive definite symmetric matrix. Prove that for any vector x,

$$\frac{(x^T x)^2}{(x^T Q x)(x^T Q^{-1} x)} \ge \frac{4\lambda_n \lambda_1}{(\lambda_n + \lambda_1)^2},$$

where λ_n and λ_1 are, respectively, the largest and smallest eigenvalues of Q. (This relation, which is known as the Kantorovich inequality, can be used to deduce (3.29) from (3.28).

3.9 Program the BFGS algorithm using the line search algorithm described in this chapter that implements the strong Wolfe conditions. Have the code verify that $y_k^T s_k$ is always positive. Use it to minimize the Rosenbrock function using the starting points given in Exercise 1.

3.10 Show that the quadratic function that interpolates $\phi(0)$, $\phi'(0)$, and $\phi(\alpha_0)$ is given by (3.41). Then, make use of the fact that the sufficient decrease condition (3.6a) is not satisfied at α_0 to show that this quadratic has positive curvature and that the minimizer satisfies

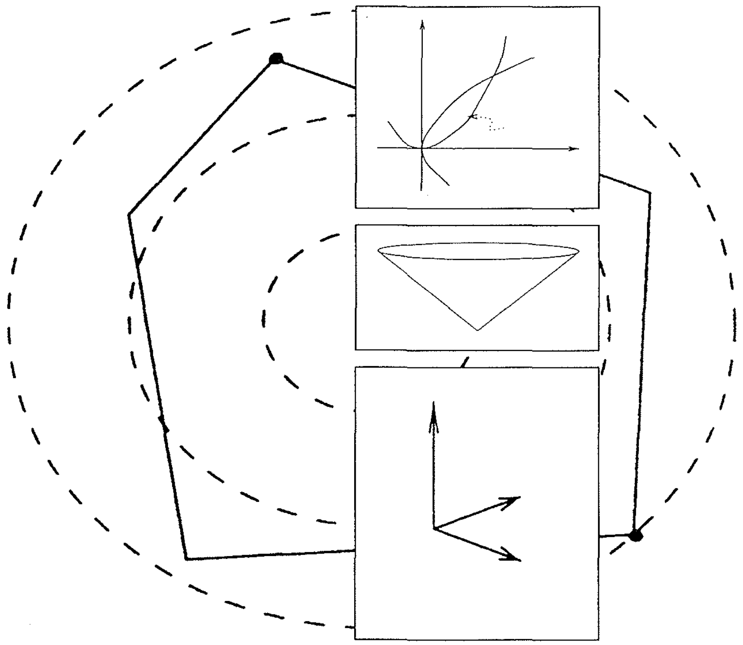
$$\alpha_1 < \frac{1}{2(1 - c_1)}.$$

Since c_1 is chosen to be quite small in practice, this indicates that α_1 cannot be much greater than $\frac{1}{2}$ (and may be smaller), which gives us an idea of the new step length.

3.11 If $\phi(\alpha_0)$ is large, (3.42) shows that α_1 can be quite small. Give an example of a function and a step length α_0 for which this situation arises. (Drastic changes to the estimate of the step length are not desirable, since they indicate that the current interpolant does not provide a good approximation to the function and that it should be modified before being trusted to produce a good step length estimate. In practice, one imposes a lower bound—typically, $\rho = 0.1$—and defines the new step length as $\alpha_i = \max(\rho\alpha_{i-1}, \hat{\alpha}_i)$, where $\hat{\alpha}_i$ is the minimizer of the interpolant.)

3.12 Suppose that the sufficient decrease condition (3.6a) is not satisfied at the step lengths α_0, and α_1, and consider the cubic interpolating $\phi(0)$, $\phi'(0)$, $\phi(\alpha_0)$ and $\phi(\alpha_1)$. By drawing graphs illustrating the two situations that can arise, show that the minimizer of the cubic lies in $[0, \alpha_1]$. Then show that if $\phi(0) < \phi(\alpha_1)$, the minimizer is less than $\frac{2}{3}\alpha_1$.

CHAPTER 5



Conjugate Gradient Methods

Our interest in the conjugate gradient method is twofold. It is one of the most useful techniques for solving large linear systems of equations, and it can also be adapted to solve nonlinear optimization problems. These two variants of the fundamental approach, which we refer to as the *linear* and *nonlinear* conjugate gradient methods, respectively, have remarkable properties that 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 very well suited for solving large problems. The performance of the linear conjugate gradient method is tied to 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

$$Ax = b, \tag{5.1}$$

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

$$\phi(x) = \tfrac{1}{2}x^T Ax - b^T x, \tag{5.2}$$

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 minimization of convex quadratic functions. For future reference we note that the gradient of ϕ equals the residual of the linear system,

$$\nabla\phi(x) = Ax - b \overset{\text{def}}{=} r(x). \tag{5.3}$$

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

$$p_i^T A p_j = 0, \qquad \text{for all } i \neq j. \tag{5.4}$$

It is easy to show that any set of vectors satisfying this property is also linearly independent.

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

$$x_{k+1} = x_k + \alpha_k p_k, \tag{5.5}$$

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

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}; \tag{5.6}$$

see (3.38). 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 (5.5), (5.6) 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:

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

for some choice of scalars σ_k. By premultiplying this expression by $p_k^T A$ and using the conjugacy property (5.4), we obtain

$$\sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k}. \tag{5.7}$$

We now establish the result by showing that these coefficients σ_k coincide with the step lengths α_k generated by the formula (5.6).

If x_k is generated by algorithm (5.5), (5.6), 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 (5.6) and (5.7), we find that $\sigma_k = \alpha_k$, giving the result. □

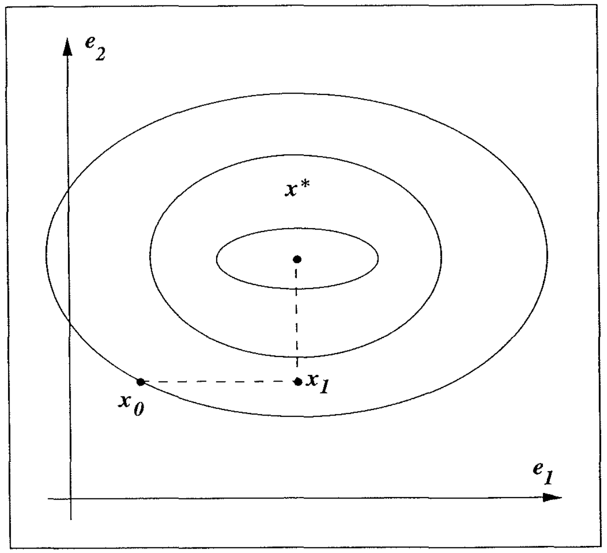


Figure 5.1 Successive minimizations along the coordinate directions find the minimizer of a quadratic with a diagonal Hessian in n iterations.

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 $e_1, e_2, \ldots, e_n$ 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 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

$$\hat{x} = S^{-1}x, \tag{5.8}$$

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

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

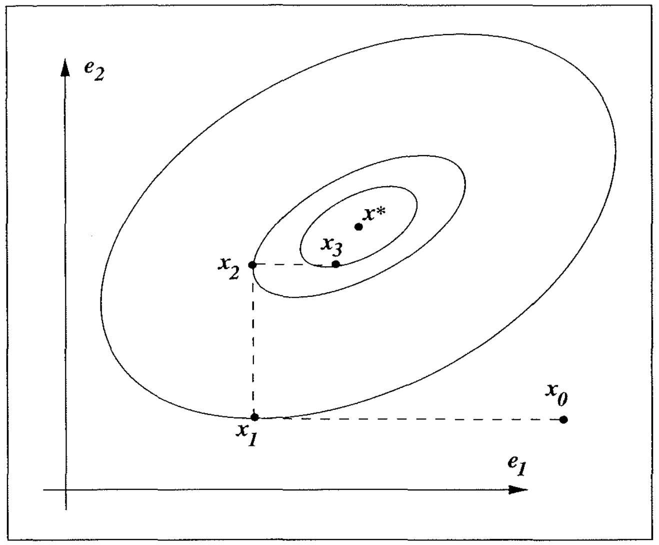


Figure 5.2 Successive minimization along coordinate axes does not find the solution in n iterations, for a general convex quadratic.

where $\{p_0, p_2, \ldots, p_{n-1}\}$ is the set of conjugate directions with respect to A. The quadratic ϕ defined by (5.2) now becomes

$$\hat{\phi}(\hat{x}) \overset{\text{def}}{=} \phi(S\hat{x}) = \tfrac{1}{2}\hat{x}^T(S^TAS)\hat{x} - (S^Tb)^T\hat{x}.$$

By the conjugacy property (5.4), the matrix S^TAS 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 (5.8), however, each 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 (5.5), (5.6). 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 $e_1, e_2, \ldots, e_k$. 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 span$\{p_0, p_1, \ldots, p_k\}$ to denote the set of all linear combinations of $p_0, p_1, \ldots, p_k$.) In proving the result we will make use of the following expression, which is easily verified from the relations (5.3) and (5.5):

$$r_{k+1} = r_k + \alpha_k A p_k. \tag{5.9}$$

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 (5.5), (5.6). Then

$$r_k^T p_i = 0, \qquad \text{for } i = 0, \ldots, k-1, \tag{5.10}$$

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

$$\{x \mid x = x_0 + \text{span}\{p_0, p_1, \ldots, p_{k-1}\}\}. \tag{5.11}$$

PROOF. We begin by showing that a point $\tilde{x}$ minimizes ϕ over the set (5.11) if and only if $r(\tilde{x})^T p_i = 0$, for each $i = 0, 1, \ldots, k-1$. Let us define $h(\sigma) = \phi(x_0 + \sigma_0 p_0 + \cdots + \sigma_{k-1}p_{k-1})$, where $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_{k-1})^T$. Since $h(\sigma)$ is a strictly convex quadratic, it has a unique minimizer σ^* that satisfies

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

By the chain rule, this implies that

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

By recalling the definition (5.3), we obtain the desired result.

We now use induction to show that x_k satisfies (5.10). Since α_k is always the one-dimensional minimizer, we have immediately that $r_1^T p_0 = 0$. Let us now make the induction hypothesis, namely, that $r_{k-1}^T p_i = 0$ for $i = 0, \ldots, k-2$. By (5.9),

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

we have

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

by the definition (5.6) of α_{k-1}. Meanwhile, for the other vectors p_i, $i = 0, 1, \ldots, k-2$, we have

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

by the induction hypothesis and the conjugacy of the p_i. We conclude that $r_k^T p_i = 0$, for $i = 0, 1, \ldots, k-1$, so that the proof is complete. □

The fact that the current residual r_k is orthogonal to all previous search directions, as expressed in (5.10), 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 (5.5), (5.6) based on *any* choice of the conjugate direction set $\{p_0, p_1, \ldots, p_{n-1}\}$. There are many ways to choose the set of conjugate directions. For instance, the eigenvectors $v_1, v_2, \ldots, v_n$ of A are mutually orthogonal as well as conjugate with respect to A, so these could be used as the vectors $\{p_0, p_1, \ldots, p_{n-1}\}$. For large-scale applications, however, it is not practical to compute the complete set of eigenvectors, for this requires a large amount of computation. An alternative 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.) This 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 $p_0, p_1, \ldots, p_{k-2}$ of the conjugate set; p_k is automatically conjugate to these vectors. This remarkable property implies that the method requires little storage and computation.

Now for the details of the conjugate gradient method. Each direction p_k is chosen to be a linear combination of the steepest descent direction $-\nabla\phi(x_k)$ (which is the same as the negative residual $-r_k$, by (5.3)) and the previous direction p_{k-1}. We write

$$p_k = -r_k + \beta_k p_{k-1}, \tag{5.12}$$

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 (5.12) by $p_{k-1}^T A$ and imposing the condition $p_{k-1}^T A p_k = 0$, we find that

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

It makes intuitive sense to 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 $r_k \neq 0$
    $\alpha_k \leftarrow -\frac{r_k^T p_k}{p_k^T A p_k}$;          (5.13a)
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$;                       (5.13b)
    $r_{k+1} \leftarrow A x_{k+1} - b$;                             (5.13c)
    $\beta_{k+1} \leftarrow \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$;  (5.13d)
    $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$;               (5.13e)
    $k \leftarrow k + 1$;                                           (5.13f)
end (while)
```

Later, we will present a more efficient version of the conjugate gradient method; the version above is useful for studying the essential properties of the method. We show first that the directions $p_0, p_1, \ldots, p_{n-1}$ 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

$$\mathcal{K}(r_0; k) \stackrel{\text{def}}{=} \text{span}\{r_0, Ar_0, \ldots, A^k r_0\}. \tag{5.14}$$

Theorem 5.3.

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

$$r_k^T r_i = 0, \qquad \text{for } i = 0, \ldots, k-1, \tag{5.15}$$

$$\text{span}\,\{r_0, r_1, \ldots, r_k\} = \text{span}\,\{r_0, Ar_0, \ldots, A^k r_0\}, \tag{5.16}$$

$$\text{span}\,\{p_0, p_1, \ldots, p_k\} = \text{span}\,\{r_0, Ar_0, \ldots, A^k r_0\}, \tag{5.17}$$

$$p_k^T A p_i = 0, \qquad \text{for } i = 0, 1, \ldots, k-1. \tag{5.18}$$

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

PROOF. The proof is by induction. The expressions (5.16) and (5.17) hold trivially for $k = 0$, while (5.18) 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 (5.16), 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 (5.16) and (5.17) that

$$r_k \in \text{span}\,\{r_0, Ar_0, \ldots, A^k r_0\}, \qquad p_k \in \text{span}\,\{r_0, Ar_0, \ldots, A^k r_0\},$$

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

$$Ap_k \in \text{span}\,\{Ar_0, \ldots, A^{k+1} r_0\}. \tag{5.19}$$

By applying (5.9), we find that

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

By combining this expression with the induction hypothesis for (5.16), we conclude that

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

To prove that the reverse inclusion holds as well, we use the induction hypothesis on (5.17) to deduce that

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

Since by (5.9) we have $Ap_i = (r_{i+1} - r_i)/\alpha_i$ for $i = 0, 1, \ldots, k$, it follows that

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

By combining this expression with the induction hypothesis for (5.16), we find that

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

Therefore, the relation (5.16) continues to hold when k is replaced by $k + 1$, as claimed.

We show that (5.17) continues to hold when k is replaced by $k + 1$ by the following argument:

$$\begin{aligned}
&\text{span}\{p_0, p_1, \ldots, p_k, p_{k+1}\} \\
&\quad = \text{span}\{p_0, p_1, \ldots, p_k, r_{k+1}\} && \text{by (5.13e)} \\
&\quad = \text{span}\{r_0, Ar_0, \ldots, A^k r_0, r_{k+1}\} && \text{by induction hypothesis for (5.17)} \\
&\quad = \text{span}\{r_0, r_1, \ldots, r_k, r_{k+1}\} && \text{by (5.16)} \\
&\quad = \text{span}\{r_0, Ar_0, \ldots, A^{k+1} r_0\} && \text{by (5.16) for } k+1.
\end{aligned}$$

Next, we prove the conjugacy condition (5.18) with k replaced by $k+1$. By multiplying (5.13e) by Ap_i, $i = 0, 1, \dots, k$, we obtain

$$p_{k+1}^T A p_i = -r_{k+1}^T A p_i + \beta_{k+1} p_k^T A p_i. \tag{5.20}$$

By the definition (5.13d) of β_k, the right-hand-side of (5.20) vanishes when $i = k$. For $i \le k-1$ we need to collect a number of observations. Note first that our induction hypothesis for (5.18) implies that the directions $p_0, p_1, \dots, p_k$ are conjugate, so we can apply Theorem 5.2 to deduce that

$$r_{k+1}^T p_i = 0, \qquad \text{for } i = 0, 1, \dots, k. \tag{5.21}$$

Second, by repeatedly applying (5.17), we find that for $i = 0, 1, \dots, k-1$, the following inclusion holds:

$$\begin{aligned} Ap_i \in A\,\text{span}\{r_0, Ar_0, \dots, A^i r_0\} &= \text{span}\{Ar_0, A^2 r_0, \dots, A^{i+1} r_0\} \\ &\subset \text{span}\{p_0, p_1, \dots, p_{i+1}\}. \end{aligned} \tag{5.22}$$

By combining (5.21) and (5.22), we deduce that

$$r_{k+1}^T A p_i = 0, \qquad \text{for } i = 0, 1, \dots, k-1,$$

so the first term in the right-hand-side of (5.20) vanishes for $i = 0, 1, \dots, k-1$. Because of the induction hypothesis for (5.18), 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 (5.18) 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 (5.15) by a noninductive argument. Because the direction set is conjugate, we have from (5.10) 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 (5.13e), 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 $i = 1, \dots, k-1$, as claimed. □

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 (5.13e) and (5.10) to replace the formula (5.13a) for α_k by

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

Second, we have from (5.9) that $\alpha_k A p_k = r_{k+1} - r_k$, so by applying (5.13e) and (5.10) 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 (5.9), 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 $r_k \neq 0$
    $\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}$;          (5.23a)
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$;          (5.23b)
    $r_{k+1} \leftarrow r_k + \alpha_k A p_k$;          (5.23c)
    $\beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$;          (5.23d)
    $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$;          (5.23e)
    $k \leftarrow k + 1$;          (5.23f)
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 Ap_k, calculation of the inner products $p_k^T(Ap_k)$ 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 unlike factorization techniques, cannot produce fill in the arrays holding the matrix. The other key property is that the CG method sometimes approaches the solution very 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 show this 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 (5.23b) and (5.17), we have that

$$\begin{aligned} x_{k+1} &= x_0 + \alpha_0 p_0 + \cdots + \alpha_k p_k \\ &= x_0 + \gamma_0 r_0 + \gamma_1 A r_0 + \cdots + \gamma_k A^k r_0, \end{aligned} \tag{5.24}$$

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

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

We can now write (5.24) as

$$x_{k+1} = x_0 + P_k^*(A) r_0. \tag{5.25}$$

We will now see that among all possible methods whose first k steps are restricted to the Krylov subspace $\mathcal{K}(r_0; k)$ given by (5.14), 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

$$\|z\|_A^2 = z^T A z. \tag{5.26}$$

(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), it is easy to show that

$$\tfrac{1}{2}\|x - x^*\|_A^2 = \tfrac{1}{2}(x - x^*)^T A (x - x^*) = \phi(x) - \phi(x^*). \tag{5.27}$$

Theorem 5.2 states that x_{k+1} minimizes ϕ, and hence $\|x - x^*\|_A^2$, over the set $x_0 + \operatorname{span}\{p_0, p_1, \ldots, p_k\}$. It follows from (5.25) 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:

$$\min_{P_k} \|x_0 + P_k(A) r_0 - x^*\|_A. \tag{5.28}$$

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

$$x_{k+1} - x^* = x_0 + P_k^*(A) r_0 - x^* = [I + P_k^*(A)A](x_0 - x^*). \tag{5.29}$$

Let $0 < \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ be the eigenvalues of A, and let $v_1, v_2, \ldots, v_n$ be the corresponding orthonormal eigenvectors. Since these eigenvectors span the whole space $\mathbb{R}^n$, we can write

$$x_0 - x^* = \sum_{i=1}^n \xi_i v_i, \tag{5.30}$$

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

$$P_k(A) v_i = P_k(\lambda_i) v_i, \qquad i = 1, 2, \ldots, n.$$

By substituting (5.30) into (5.29) we have

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

and hence

$$\|x_{k+1} - x^*\|_A^2 = \sum_{i=1}^n \lambda_i [1 + \lambda_i P_k^*(\lambda_i)]^2 \xi_i^2. \tag{5.31}$$

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

$$\begin{aligned} \|x_{k+1} - x^*\|_A^2 &\le \min_{P_k} \max_{1\le i\le n} [1 + \lambda_i P_k(\lambda_i)]^2 \left(\sum_{j=1}^{n} \lambda_j \xi_j^2\right) \\ &= \min_{P_k} \max_{1\le i\le n} [1 + \lambda_i P_k(\lambda_i)]^2 \|x_0 - x^*\|_A^2, \end{aligned} \tag{5.32}$$

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

The expression (5.32) allows us to quantify the convergence rate of the CG method by estimating the nonnegative scalar quantity

$$\min_{P_k} \max_{1\le i\le n} [1 + \lambda_i P_k(\lambda_i)]^2. \tag{5.33}$$

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 $\lambda_1, \lambda_2, \ldots, \lambda_n$ take on the r distinct values $\tau_1 < \tau_2 < \cdots < \tau_r$. 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 $i = 1, 2, \ldots, n$ 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 (5.33), we have

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

Hence the constant in (5.33) is zero for the value $k = r - 1$, so we have by substituting into (5.32) that $\|x_r - x^*\|_A^2 = 0$, and therefore $x_r = x^*$, as claimed. □

By using similar reasoning, Luenberger [152] establishes the following estimate, which gives a useful characterization of the behavior of the CG method.

Theorem 5.5.

If A has eigenvalues $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$, we have that

$$\|x_{k+1} - x^*\|_A^2 \le \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\right)^2 \|x_0 - x^*\|_A^2. \tag{5.34}$$

Without giving details of the proof, we describe how this result is obtained from (5.32). 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 $\lambda_n, \lambda_{n-1}, \ldots, \lambda_{n-k+1}$, 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 $\lambda_1, \lambda_2, \ldots, \lambda_{n-k}$ 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

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

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.

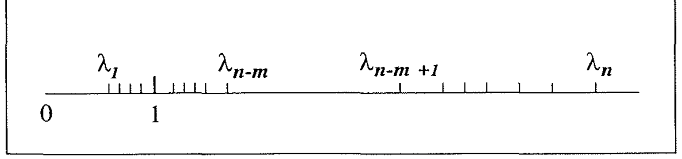


Figure 5.3 Two clusters of eigenvalues.

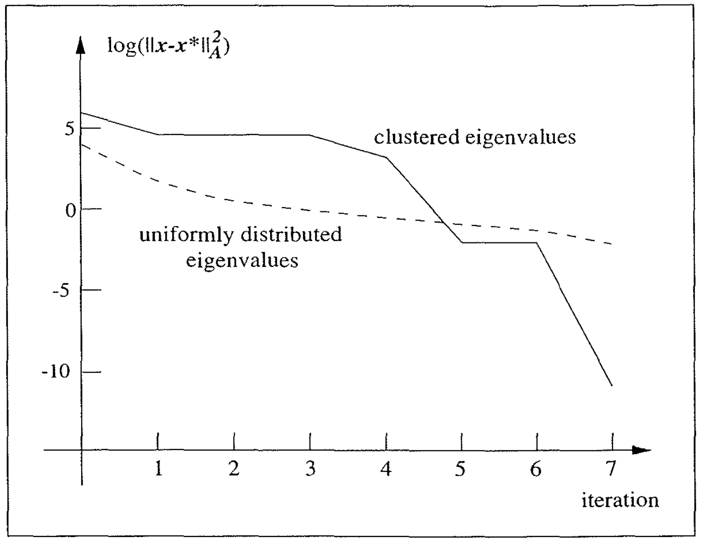


Figure 5.4 Performance of the conjugate gradient method on (a) a problem in which five of the eigenvalues are large and the remainder are clustered near 1, and (b) a matrix with uniformly distributed eigenvalues.

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 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 the next iteration, i.e. iteration 7.

To state this more precisely, it is generally true that if the eigenvalues occur in r distinct clusters, the CG iterates will *approximately* solve the problem after r steps (see [115]). This result can be proved by constructing a polynomial $\bar{P}_{r-1}$ such that $(1 + \lambda\bar{P}_{r-1}(\lambda))$ has zeros inside each of the clusters. This polynomial may not vanish at the eigenvalues λ_i, $i = 1, 2, \ldots, n$, but its value will be small at these points, so the constant defined in (5.33) will be tiny for $k \geq r - 1$. We illustrate this behavior in Figure 5.5, which shows the

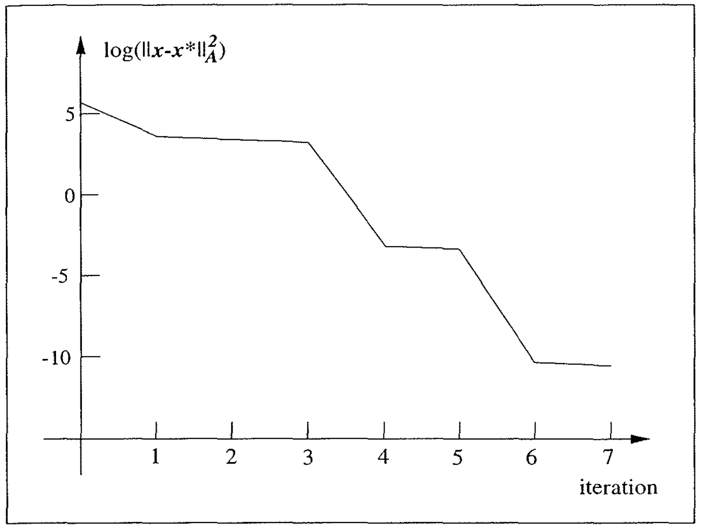


Figure 5.5 Performance of the conjugate gradient method on a matrix in which the eigenvalues occur in four distinct clusters.

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 norm is quite small. 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

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

It can be shown that

$$\|x_k - x^*\|_A \leq \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^{2k} \|x_0 - x^*\|_A. \tag{5.35}$$

This bound often gives a large overestimate of the error, but it can be useful in those cases 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 (3.28), 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,

$$\hat{x} = Cx. \tag{5.36}$$

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

$$\hat{\phi}(\hat{x}) = \tfrac{1}{2}\hat{x}^T(C^{-T}AC^{-1})\hat{x} - (C^{-T}b)^T\hat{x}. \tag{5.37}$$

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 (5.35) 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 (5.36) explicitly. Rather, we can apply Algorithm 5.2 to the problem (5.37), 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 $M = C^TC$, 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 $p_0 = -r_0$, $k \leftarrow 0$;
while $r_k \neq 0$
    $\alpha_k \leftarrow \dfrac{r_k^T y_k}{p_k^T A p_k}$;        (5.38a)
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$;        (5.38b)
    $r_{k+1} \leftarrow r_k + \alpha_k A p_k$;        (5.38c)
    $M y_{k+1} \leftarrow r_{k+1}$;        (5.38d)
    $\beta_{k+1} \leftarrow \dfrac{r_{k+1}^T y_{k+1}}{r_k^T y_k}$;        (5.38e)
    $p_{k+1} \leftarrow -y_{k+1} + \beta_{k+1} p_k$;        (5.38f)
    $k \leftarrow k + 1$;        (5.38g)
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 (5.15) of the successive residuals becomes

$$r_i^T M^{-1} r_j = 0 \quad \text{for all } i \neq j. \tag{5.39}$$

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

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 [220], [115], and [53] for discussions of these techniques.) *Incomplete Cholesky* is probably the most effective in general; we discussed it briefly in Chapter 6. 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.

THE FLETCHER–REEVES METHOD

Fletcher and Reeves [88] showed that an extension of this kind is possible by making two simple changes in Algorithm 5.2. First, in place of the choice (5.23a) 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, 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-CG).

```
Given $x_0$;
Evaluate $f_0 = f(x_0)$, $\nabla f_0 = \nabla f(x_0)$;
Set $p_0 = -\nabla f_0$, $k \leftarrow 0$;
while $\nabla f_k \neq 0$
    Compute $\alpha_k$ and set $x_{k+1} = x_k + \alpha_k p_k$;
    Evaluate $\nabla f_{k+1}$;
    $\beta^{FR}_{k+1} \leftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$;    (5.40a)
    $p_{k+1} \leftarrow -\nabla f_{k+1} + \beta^{FR}_{k+1} p_k$;    (5.40b)
    $k \leftarrow k + 1$;    (5.40c)
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 performed, 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 (5.40b), the search direction p_k may fail to be a descent direction unless α_k satisfies certain conditions. By taking the inner product of (5.40b) (with k replacing $k+1$) with the gradient vector ∇f_k, we obtain

$$\nabla f_k^T p_k = -\|\nabla f_k\|^2 + \beta_k^{FR} \nabla f_k^T p_{k-1}. \tag{5.41}$$

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 (5.41) that $\nabla f_k^T p_k < 0$, so that p_k is indeed a descent direction. But if the line search is not exact, the second term in (5.41) 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:

$$f(x_k + \alpha_k p_k) \le f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, \tag{5.42a}$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \le c_2 |\nabla f_k^T p_k|, \tag{5.42b}$$

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 (5.42b) implies that (5.41) is negative, and we conclude that any line search procedure that yields an α_k satisfying (5.42) will ensure that all directions p_k are descent directions for the function f.

THE POLAK–RIBIÈRE METHOD

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

$$\beta_{k+1}^{\text{PR}} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}. \tag{5.43}$$

We refer to the algorithm in which (5.43) replaces (5.40a) as Algorithm PR-CG, and refer to Algorithm 5.4 as Algorithm FR-CG. They are identical when f is a strongly convex quadratic function and the line search is exact, since by (5.15) 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-CG tends to be the more robust and efficient of the two.

A surprising fact about Algorithm PR-CG is that the strong Wolfe conditions (5.42) do not guarantee that p_k is always a descent direction. If we define the β parameter as

$$\beta_{k+1}^{+} = \max\{\beta_{k+1}^{\text{PR}}, 0\}, \tag{5.44}$$

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

$$\beta_{k+1}^{\text{HS}} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{(\nabla f_{k+1} - \nabla f_k)^T p_k}, \tag{5.45}$$

gives rise to an algorithm that is similar to Algorithm PR-CG, both in terms of its theoretical convergence properties and in its practical performance. Formula (5.45) 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

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

Recalling from Taylor's theorem (2.5) 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 (5.45).

None of the other proposed definitions of β_k has proved to be significantly more efficient than the Polak–Ribière choice (5.43).

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 (5.40a), 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,

$$\|x_{k+n} - x\| = O\left(\|x_k - x^*\|^2\right). \tag{5.46}$$

With a little thought, we can see that 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 holds 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 (2.6) 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 (5.46).

Though the result (5.46) 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. In such problems restarts may never occur, since an approximate solution is often 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 (5.15), 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

$$\frac{|\nabla f_k^T \nabla f_{k-1}|}{\|\nabla f_k\|^2} \geq \nu, \tag{5.47}$$

where a typical value for the parameter ν is 0.1.

Another modification to the restart strategy is to use a direction other than steepest descent as the restart direction. The Harwell subroutine VA14 [133], for instance, defines

p_{k+1} by using a three-term recurrence based on ∇f_{k+1}, p_k and a third direction that contains earlier information about the behavior of the objective function. An algorithm that takes this idea one step further is CONMIN, which is discussed in Chapter 9.

We could also think of formula (5.44) as a restarting strategy, because p_{k+1} will revert to the steepest descent direction whenever β_k^{PR} is negative. In contrast to (5.47), however, these restarts are rather infrequent because β_k^{PR} is positive most of the time.

NUMERICAL PERFORMANCE

Table 5.1 illustrates the performance of Algorithms FR-CG, PR-CG, and PR+ without restarts. For these tests, the parameters in the strong Wolfe conditions (5.42) were chosen to be $c_1 = 10^{-4}$ and $c_2 = 0.1$. The iterations were terminated when

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

or after 10,000 iterations (the latter is denoted by a $*$).

The final column, headed "mod," indicates the number of iterations of Algorithm PR+ for which the adjustment (5.44) was needed to ensure that $\beta_k^{\text{PR}} \geq 0$. An examination of the results of Algorithm FR-CG on problem GENROS shows that the method takes very short steps far from the solution that lead to tiny improvements in the objective function.

The Polak–Ribière algorithm, or its variation PR+, are not *always* more efficient than Algorithm FR-CG, and it has the slight disadvantage of requiring one more vector of storage. Nevertheless, we recommend that users choose Algorithm PR-CG or PR+ whenever possible.

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 that guarantee that all search directions are descent directions. It assumes that the level set $\mathcal{L} = \{x : f(x) \leq f(x_0)\}$ is

Table 5.1 Iterations and function/gradient evaluations required by three nonlinear conjugate gradient methods on a set of test problems.

		Alg FR	Alg PR	Alg PR+	
Problem	n	it/f-g	it/f-g	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

bounded, and that f is twice continuously differentiable, so that we have from Lemma 3.1 that there exists a step length α_k that satisfies 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 (5.42) with $0 < c_2 < \frac{1}{2}$. Then the method generates descent directions p_k that satisfy the following inequalities:

$$-\frac{1}{1-c_2} \leq \frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2} \leq \frac{2c_2 - 1}{1 - c_2}, \quad \text{for all } k = 0, 1, \ldots. \tag{5.48}$$

PROOF. Note first that the function $t(\xi) \overset{\text{def}}{=} (2\xi - 1)(1 - \xi)$ 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

$$-1 < \frac{2c_2 - 1}{1 - c_2} < 0. \tag{5.49}$$

The descent condition $\nabla f_k^T p_k < 0$ follows immediately once we establish (5.48).

The proof is by induction. For $k = 0$, the middle term in (5.48) is -1, so by using (5.49), we see that both inequalities in (5.48) are satisfied. Next, assume that (5.48) holds for some $k \geq 1$. From (5.40b) and (5.40a) we have

$$\frac{\nabla f_{k+1}^T p_{k+1}}{\|\nabla f_{k+1}\|^2} = -1 + \beta_{k+1}\frac{\nabla f_{k+1}^T p_k}{\|\nabla f_{k+1}\|^2} = -1 + \frac{\nabla f_{k+1}^T p_k}{\|\nabla f_k\|^2}. \tag{5.50}$$

By using the line search condition (5.42b), we have

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

so by combining with (5.50), 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 (5.48), 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 (5.48) holds for $k + 1$ as well. □

This result used only the second strong Wolfe condition (5.42b); the first Wolfe condition (5.42a) will be needed in the next section to establish global convergence. The bounds on $\nabla f_k^T p_k$ in (5.48) 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

$$\cos\theta_k = \frac{-\nabla f_k^T p_k}{\|\nabla f_k\| \, \|p_k\|}. \tag{5.51}$$

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 (5.48) by $\|\nabla f_k\|/\|p_k\|$ and using (5.51), we obtain

$$\chi_1 \frac{\|\nabla f_k\|}{\|p_k\|} \le \cos\theta_k \le \chi_2 \frac{\|\nabla f_k\|}{\|p_k\|}, \qquad \text{for all } k = 0, 1, \ldots, \tag{5.52}$$

where χ_1 and χ_2 are two positive constants. From the right-hand inequality we can have $\cos\theta_k \approx 0$ 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

$$\beta_{k+1}^{\text{FR}} \approx 1, \tag{5.53}$$

by the definition (5.40a). By using this approximation together with $\|\nabla f_{k+1}\| \approx \|\nabla f_k\| \ll \|p_k\|$ in (5.40b), 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 (5.43) that $\beta_{k+1}^{\text{PR}} \approx 0$. From the formula (5.40b), 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-CG essentially performs a restart after it encounters a bad direction. The same argument can be applied to Algorithms PR+ and HS-CG.

The inefficient behavior of the Fletcher–Reeves method predicted by the arguments given above can be observed in practice. For example, the paper [103] 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-CG requires thousands of iterations to solve this problem, while Algorithm PR-CG 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-CG 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. The theory developed in the literature offers fascinating glimpses into their behavior, but our knowledge remains fragmentary. 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.

Assumption 5.1.

(i) *The level set $\mathcal{L} := \{x : f(x) \le f(x_0)\}$ is bounded.*

(ii) *In some neighborhood $\mathcal{N}$ of $\mathcal{L}$, the objective function f is Lipschitz continuously differentiable, that is, there exists a constant $L > 0$ such that*

$$\|\nabla f(x) - \nabla f(\tilde{x})\| \le L\|x - \tilde{x}\|, \qquad \text{for all } x, \tilde{x} \in \mathcal{N}. \tag{5.54}$$

This assumption implies that there is a constant $\bar{\gamma}$ such that

$$\|\nabla f(x)\| \le \bar{\gamma}, \text{ for all } x \in \mathcal{L}. \tag{5.55}$$

Our main analytical tool in this section is Zoutendijk's theorem—Theorem 3.2 in Chapter 3—which we restate here for convenience.

Theorem 5.7.

Suppose that Assumptions 5.1 hold. Consider 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 (5.42). Then

$$\sum_{k=1}^{\infty} \cos^2\theta_k \, \|\nabla f_k\|^2 < \infty. \tag{5.56}$$

We can use this theorem 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 (5.56) that

$$\sum_{k=k_1,k_2,\ldots} \|\nabla f_k\|^2 < \infty. \tag{5.57}$$

If we allow no more than $\bar{n}$ iterations between restarts, the sequence $\{k_j\}_{j=1}^{\infty}$ will be infinite, and from (5.57) we have that $\lim_{j\to\infty} \|\nabla f_{k_j}\| = 0$. That is, a subsequence of gradients approaches zero, or equivalently,

$$\liminf_{k\to\infty} \|\nabla f_k\| = 0. \tag{5.58}$$

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 Theorem 5.7 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 (as in the restarted method above), the following result shows that it is at least not bounded away from zero.

Theorem 5.8.

Suppose that Assumptions 5.1 hold, and that Algorithm 5.4 is implemented with a line search that satisfies the strong Wolfe conditions (5.42), with $0 < c_1 < c_2 < \frac{1}{2}$. *Then*

$$\liminf_{k\to\infty} \|\nabla f_k\| = 0. \tag{5.59}$$

PROOF. The proof is by contradiction. It assumes that the opposite of (5.59) holds, that is, there is a constant $\gamma > 0$ such that

$$\|\nabla f_k\| \geq \gamma, \qquad \text{for all } k \text{ sufficiently large,} \tag{5.60}$$

and uses Lemma 5.6 and Theorem 5.7 to derive the contradiction.

From Lemma 5.6, we have that

$$\cos\theta_k \geq \frac{1}{1-c_2}\frac{\|\nabla f_k\|}{\|p_k\|}, \qquad k = 1, 2, \ldots, \tag{5.61}$$

and by substituting this relation in Zoutendijk's condition (5.56), we obtain

$$\sum_{k=1}^{\infty} \frac{\|\nabla f_k\|^4}{\|p_k\|^2} < \infty. \tag{5.62}$$

By using (5.42b) and Lemma 5.6 again, we obtain that

$$|\nabla f_k^T p_{k-1}| \leq -c_2 \nabla f_{k-1}^T p_{k-1} \leq \frac{c_2}{1-c_2}\|\nabla f_{k-1}\|^2. \tag{5.63}$$

Thus, from (5.40b) and recalling the definition (5.40a) of β_k^{FR} we obtain

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

Applying this relation repeatedly, defining $c_3 \overset{\text{def}}{=} (1+c_2)/(1-c_2) \geq 1$, and using the definition (5.40a) of β_k^{FR}, we have

$$\begin{aligned}
\|p_k\|^2 &\leq c_3\|\nabla f_k\|^2 + (\beta_k^{\text{FR}})^2[\chi_3\|\nabla f_{k-1}\|^2 + (\beta_{k-1}^{\text{FR}})^2\|p_{k-2}\|^2] \\
&= c_3\|\nabla f_k\|^4\left[\frac{1}{\|\nabla f_k\|^2} + \frac{1}{\|\nabla f_{k-1}\|^2}\right] + \frac{\|\nabla f_k\|^4}{\|\nabla f_{k-2}\|^4}\|p_{k-2}\|^2 \\
&\leq c_3\|\nabla f_k\|^4 \sum_{j=1}^{k} \|\nabla f_j\|^{-2},
\end{aligned} \tag{5.64}$$

where we used the facts that

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

and $p_1 = -\nabla f_1$. By using the bounds (5.55) and (5.60) in (5.64), we obtain

$$\|p_k\|^2 \leq \frac{\chi_3\bar{\gamma}^4}{\gamma^2}k, \tag{5.65}$$

which implies that

$$\sum_{k=1}^{\infty} \frac{1}{\|p_k\|^2} \geq \gamma_4 \sum_{k=1}^{\infty} \frac{1}{k}, \tag{5.66}$$

for some positive constant γ_4.

Suppose for contradiction that (5.60) holds. Then from (5.62), we have that

$$\sum_{k=1}^{\infty} \frac{1}{\|p_k\|^2} < \infty. \tag{5.67}$$

However, if we combine this inequality with (5.66), we obtain that $\sum_{k=1}^{\infty} 1/k < \infty$, which is not true. Hence, (5.60) does not hold, and the claim (5.59) is proved. □

Note that this global convergence result applies to a practical implementation of the Fletcher–Reeves method and is valid for general nonlinear objective functions. In this sense, it is more satisfactory than other convergence results that apply to specific types of objectives—for example, convex functions.

In general, if we can show that there is a constant $c_4 > 0$ such that

$$\cos\theta_k \geq c_4 \frac{\|\nabla f_k\|}{\|p_k\|}, \qquad k = 1, 2, \ldots,$$

and another constant c_5 such that

$$\frac{\|\nabla f_k\|}{\|p_k\|} \geq c_5 > 0, \qquad k = 1, 2, \ldots,$$

it follows from Theorem 5.7 that

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

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, is it not possible to prove a result like Theorem 5.8 for Algorithm PR-CG. This is unexpected, since the Polak–Ribière method performs better in practice than the Fletcher–Reeves method. In fact, 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 stationary point for the function $t(\alpha) = f(x_k + \alpha p_k)$.)

Theorem 5.9.

Consider the Polak–Ribière method method (5.43), with an ideal line search. There exists a twice continuously differentiable objective function $f : \mathbb{R}^3 \to \mathbb{R}$ 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 is given in [207], and 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 5.9 requires that some consecutive search directions become almost negatives of each other. In the case of ideal line searches, this can be achieved only if $\beta_k < 0$, so the analysis suggests a modification of the Polak–Ribière method in which we set

$$\beta_k^{+} = \max\{\beta_k^{\text{PR}}, 0\}. \tag{5.68}$$

This method is exactly Algorithm PR+ discussed above. 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 global convergence of Algorithm PR+ for general functions.

NOTES AND REFERENCES

The conjugate gradient method was developed in the 1950s by Hestenes and Stiefel [135] as an alternative to factorization methods for finding exact 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 [152].

Interestingly enough, the nonlinear conjugate gradient method of Fletcher and Reeves [88] was proposed after the linear conjugate gradient method had fallen out of favor, but several years before it was rediscovered as an iterative method. The Polak–Ribière method was proposed in [194], and the example showing that it may fail to converge on nonconvex problems is given by Powell [207]. Restart procedures are discussed in Powell [203].

Analysis due to Powell [200] 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

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

then the angle between the gradient ∇f_k and the search direction p_k stays constant. Therefore, if this angle is close to 90°, the method will converge very slowly. Indeed, 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 is studied also in Al-Baali [3] and Gilbert and Nocedal [103].

Most of the theory on the *rate of convergence* of conjugate gradient methods assumes that the line search is exact. Crowder and Wolfe [61] show that the rate of convergence is linear, and show by constructing an example that Q-superlinear convergence is not achievable. Powell [200] 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 [43] and Burmeister [33] prove n-step quadratic convergence for general objective functions, that is,

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

Ritter [214] shows that in fact, the rate is *superquadratic*, that is,

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

Powell [206] 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 [6] and Stoer [232].

Even faster rates of convergence can be established (see Schuller [225], Ritter [214]), 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 [180] 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 [180] presented an algorithm that attains this optimal bound. It is related to PARTAN, the method of parallel tangents (see, for example, Luenberger [152]). 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.

Special line search strategies that ensure global convergence of the Polak–Ribière method have been proposed, but they are not without disadvantages.

The results in Table 5.1 are taken from Gilbert and Nocedal [103]. This paper also describes a line search that guarantees that Algorithm PR+ always generates descent directions, and proves global convergence.

EXERCISES

5.1 Implement Algorithm 5.2 and use to it 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 $b = (1, 1, \ldots, 1)^T$

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 $p_0, p_1, \ldots, p_l$ satisfy (5.4), 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 (5.6).

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

5.5 Using the form of the CG iteration prove directly that (5.16) and (5.17) hold for $k = 1$.

5.6 Show that (5.23d) is equivalent to (5.13d).

5.7 Let $\{\lambda_i, v_i\}$ $i = 1, \ldots, n$ be the eigenpairs of 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 and apply the CG method to them. Then observe 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 (5.39).

5.11 Show that when applied to a quadratic function, and with exact line searches, the Polak–Ribière formula given by (5.43) and the Hestenes–Stiefel formula given by (5.45) reduce to the Fletcher–Reeves formula (5.40a).

5.12 Prove that Lemma 5.6 holds for any choice of β_k satisfying $|\beta_k| \le \beta_k^{\text{FR}}$.