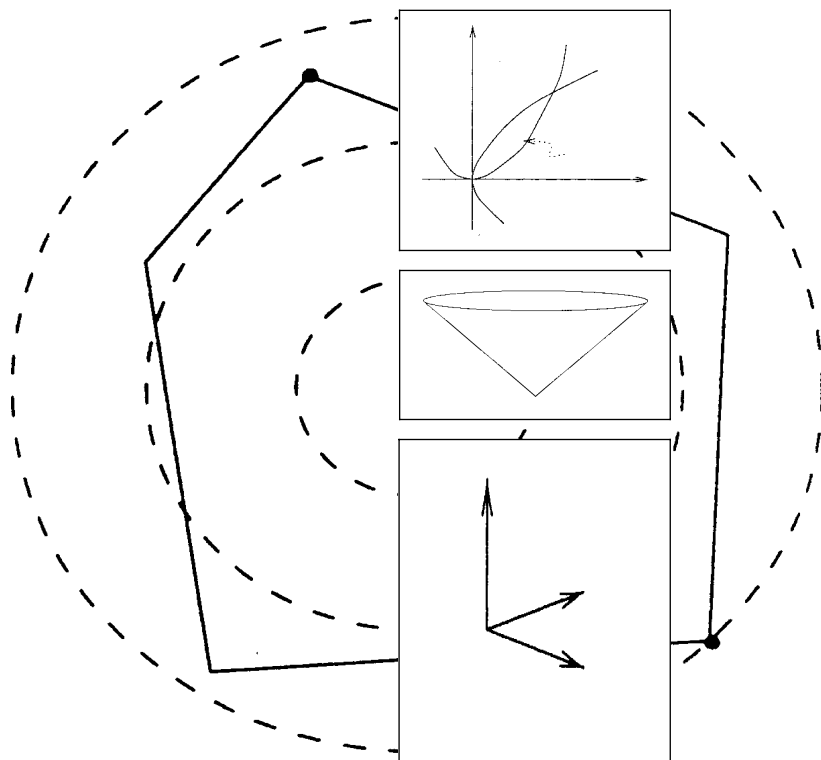


CHAPTER 2



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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$ 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, \dots . 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.

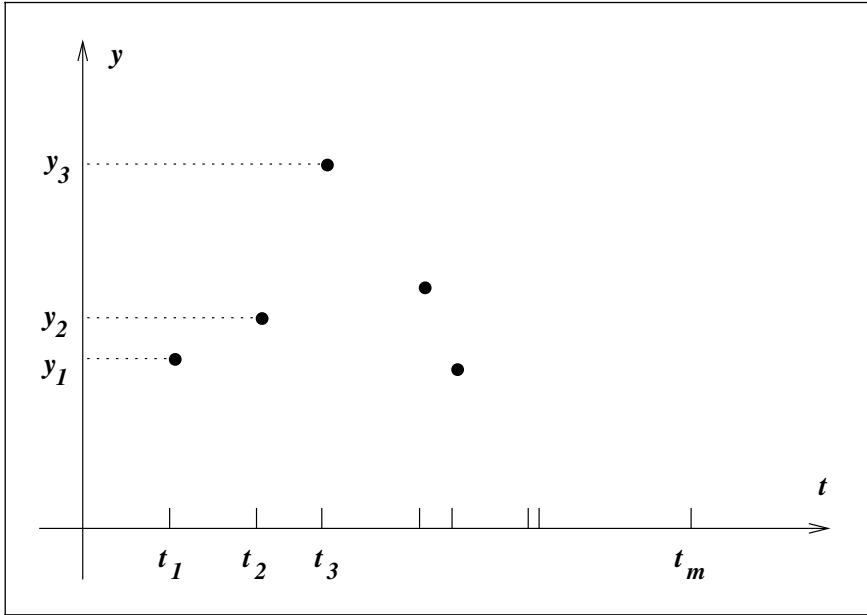


Figure 2.1 Least squares data fitting problem.

□ 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, \dots, y_m of a signal taken at times t_1, t_2, \dots, 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 , $i = 1, 2, \dots, 6$, 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 $x = (x_1, x_2, \dots, x_6)^T$, and define the residuals

$$r_j(x) = y_j - \phi(t_j; x), \quad j = 1, \dots, m, \quad (2.2)$$

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

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

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. \square

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

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

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 $x_n \rightarrow 0$ as $n \rightarrow \infty$.

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.

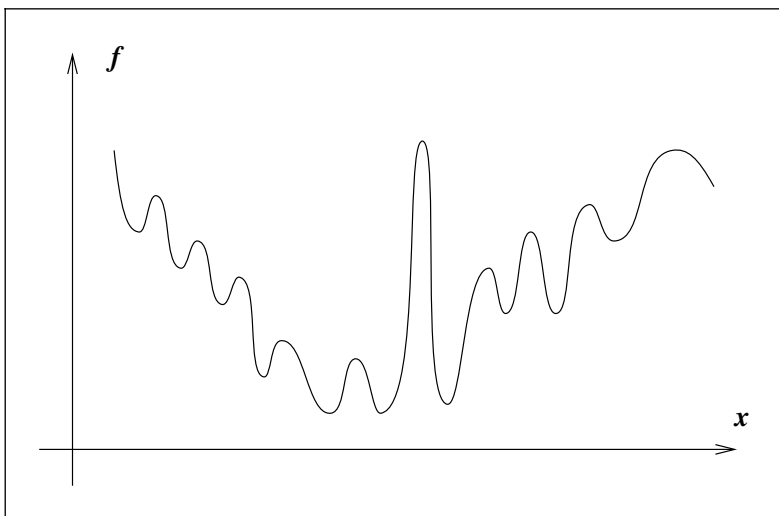


Figure 2.2 A difficult case for global minimization.

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

$$f(x + p) = f(x) + \nabla f(x + tp)^T p, \quad (2.4)$$

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

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

and that

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

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

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

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

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

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. \square

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 $p^T B p > 0$ for all $p \neq 0$, and positive semidefinite if $p^T B p \geq 0$ 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

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

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. \square

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

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

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. \square

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,

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

By the convexity property for f , we have

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

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

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

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

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 examining 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

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

(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].

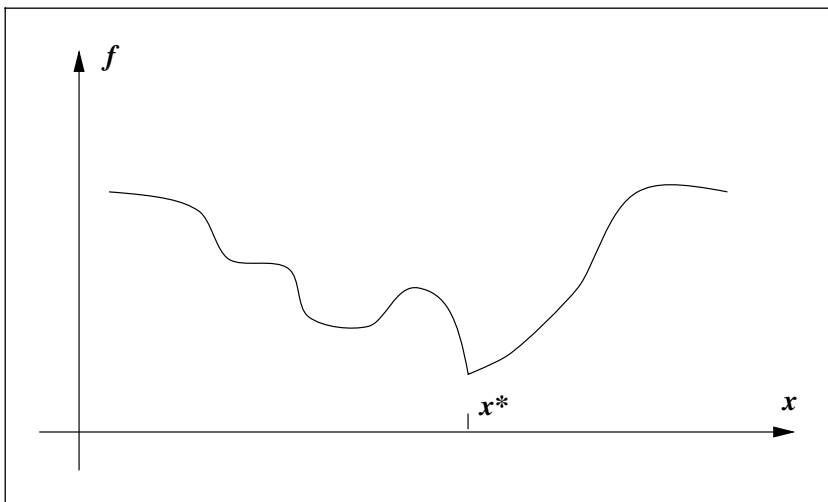


Figure 2.3 Nonsmooth function with minimum at a kink.

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, \dots, 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 :

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

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:

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

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 $\|p\|_2 \leq \Delta$, 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

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

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

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

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.

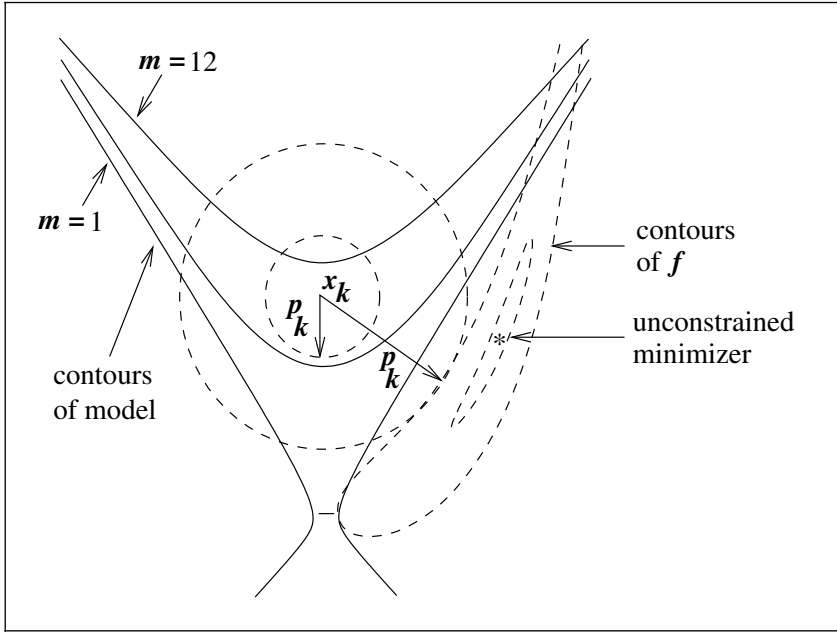


Figure 2.4 Two possible trust regions (circles) and their corresponding steps p_k . The solid lines are contours of the model function m_k .

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

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

(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

$$\min_p p^T \nabla f_k, \quad \text{subject to } \|p\| = 1. \quad (2.12)$$

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 $\cos \theta$

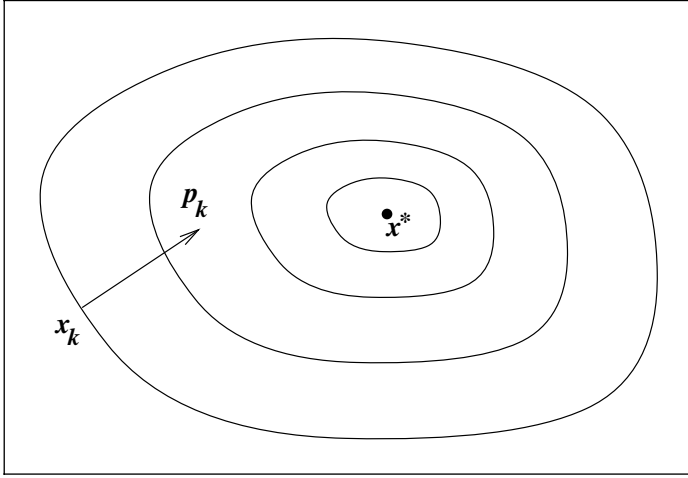


Figure 2.5 Steepest descent direction for a function of two variables.

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

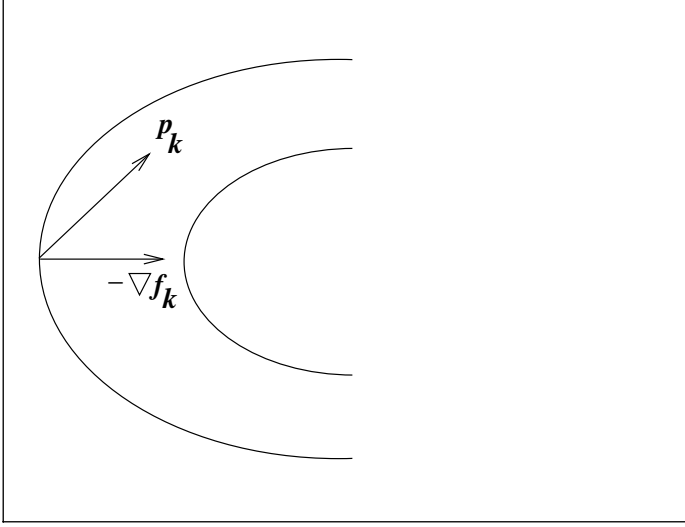


Figure 2.6 A downhill direction p_k

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

$$f(x_k + p) \approx f_k + p^T \nabla f_k + \frac{1}{2} p^T \nabla^2 f_k p \stackrel{\text{def}}{=} m_k(p). \quad (2.13)$$

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:

$$p_k^N = -\nabla^2 f_k^{-1} \nabla f_k. \quad (2.14)$$

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

$$\nabla f_k^T p_k^N = -p_k^{NT} \nabla^2 f_k p_k^N \leq -\sigma_k \|p_k^N\|^2$$

for some $\sigma_k > 0$. Unless the gradient ∇f_k (and therefore the step p_k^N) is zero, we have that $\nabla f_k^T p_k^N < 0$, 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 $\nabla f_k^T p_k^N < 0$, 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

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

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

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

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*:

$$B_{k+1}s_k = y_k, \quad (2.16)$$

where

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

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

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

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

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

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,

$$p_k = -B_k^{-1} \nabla f_k. \quad (2.19)$$

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 $H_k \stackrel{\text{def}}{=} B_k^{-1}$, is

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

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

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

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

$$\min_p f_k + p^T \nabla f_k \quad \text{subject to } \|p\|_2 \leq \Delta_k.$$

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 $\|p\|_2 \leq \Delta_k$, 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:

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

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

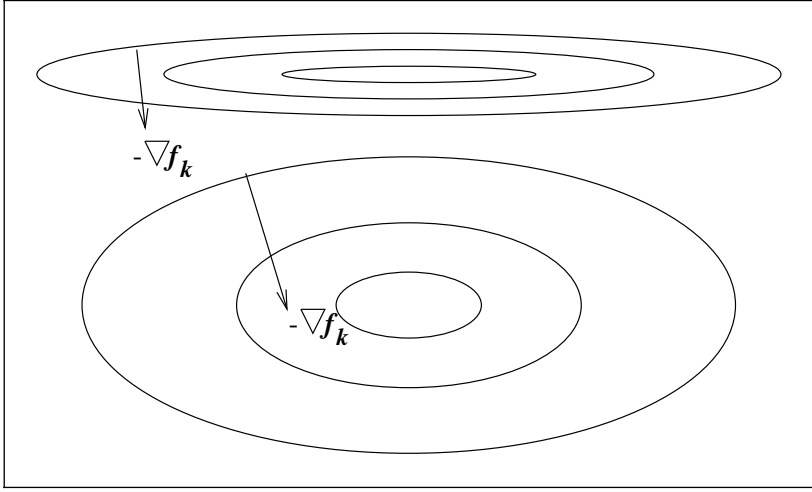


Figure 2.7 Poorly scaled and well-scaled problems, and performance of the steepest descent direction.

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

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

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

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

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

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

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

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

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 $\{v_k\}$ such that

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

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

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

(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

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

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 $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ 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 $k = 0, 1, 2, \dots$. 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


$$\xi_k = k \pmod{2\pi} = k - 2\pi \left\lfloor \frac{k}{2\pi} \right\rfloor,$$

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 $f(x_1, x_2) = (x_1 + x_2^2)^2$. 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

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

(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 $i, j = 1, 2, \dots, n$.)

 **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?