3.15. Let $g_1, \ldots, g_m$ be concave functions on $\Re^n$. Prove that the set

$$S = \{ x : g_i(x) \geq 0, i = 1, \ldots, m \}$$

is convex.

3.16. Let $f : \Re^n \to \Re^1$ be a convex function, and let $g : \Re^1 \to \Re^1$ be a convex nondecreasing function. (The notation $f : \Re^n \to \Re^1$ means that f is a real-valued function of n variables; g is a real-valued function of one variable.) Prove that the composite function $h : \Re^n \to \Re^1$ defined by $h(x) = g(f(x))$ is convex.

3.17. Complete the proof of Theorem 2.1 for the case when the objective function is strictly convex.

3.18. Express $(2, 2)^T$ as a convex combination of $(0, 0)^T$, $(1, 4)^T$, and $(3, 1)^T$.

3.19. For each of the following functions, determine if it is convex, concave, both, or neither on the real line. If the function is convex or concave, indicate if it is strictly convex or strictly concave.

(i) $f(x) = 3x^2 + 4x - 5$
(ii) $f(x) = \exp(x^2)$
(iii) $f(x) = 7x - 15$
(iv) $f(x) = \sqrt{1 + x^2}$
(v) $f(x) = 4 - 5x + 3x^2$
(vi) $f(x) = 2x^4 + 3x^3 + 4x^2$
(vii) $f(x) = x/(1 + x^4)$.

3.20. Determine if

$$f(x_1, x_2) = 2x_1^2 - 3x_1x_2 + 5x_2^2 - 2x_1 + 6x_2$$

is convex, concave, both, or neither for $x \in \Re^2$.

3.21. Give an example of a one-dimensional function f that is strictly convex on the real line even though $f''(\hat{x}) = 0$ at some point $\hat{x}$.

3.22. Let $g_1, \ldots, g_m$ be concave functions on $\Re^n$, let f be a convex function on $\Re^n$, and let μ be a positive constant. Prove that the function

$$\beta(x) = f(x) - \mu \sum_{i=1}^{m} \log g_i(x)$$

is convex on the set $S = \{ x : g_i(x) > 0, i = 1, \ldots, m \}$.

2.4 The General Optimization Algorithm

More algorithms for solving optimization problems have been proposed than could possibly be discussed in a single book. This has happened in part because optimization problems can come in so many forms, but even for particular problems such as one-variable unconstrained minimization problems, there are many different algorithms that one could use.

Despite this diversity of both algorithms and problems, all of the algorithms that we will discuss in any detail in this book will have the same general form.

ALGORITHM 2.1.
General Optimization Algorithm I

1. Specify some initial guess of the solution x_0.
2. For $k = 0, 1, \ldots$
   (i) If x_k is optimal, stop.
   (ii) Determine x_{k+1}, a new estimate of the solution.

This algorithm is so simple that it almost conveys no information at all. However, as we discuss ever more complex algorithms for ever more elaborate problems, it is often helpful to keep in mind that we are still working within this simple and general framework.

The algorithm suggests that testing for optimality and determining a new point x_{k+1} are separate ideas, but this is usually not true. Often the information obtained from the optimality test is the basis for the computation of the new point. For example, if we are trying to solve the one-dimensional problem without constraints

$$\text{minimize } f(x),$$

then the optimality test will often be based on the condition

$$f'(x) = 0.$$

If $f'(x_k) \neq 0$, then x_k is not optimal, and the sign and value of $f'(x_k)$ indicate whether f is increasing or decreasing at the point x_k, as well as how rapidly f is changing. Such information is valuable in selecting x_{k+1}.

Many of our algorithms will have a more specific form.

ALGORITHM 2.2.
General Optimization Algorithm II

1. Specify some initial guess of the solution x_0.
2. For $k = 0, 1, \ldots$
   (i) If x_k is optimal, stop.
   (ii) Determine a *search direction* p_k.
   (iii) Determine a *step length* α_k that leads to an improved estimate of the solution: $x_{k+1} = x_k + \alpha_k p_k$.

In this algorithm, p_k is a *search direction* that we hope points in the general direction of the solution, or that "improves" our solution in some sense. The scalar α_k is a *step length* that determines the point x_{k+1}; once the search direction p_k has been computed, the step length α_k is found by solving some auxiliary one-dimensional problem; see Figure 2.7.

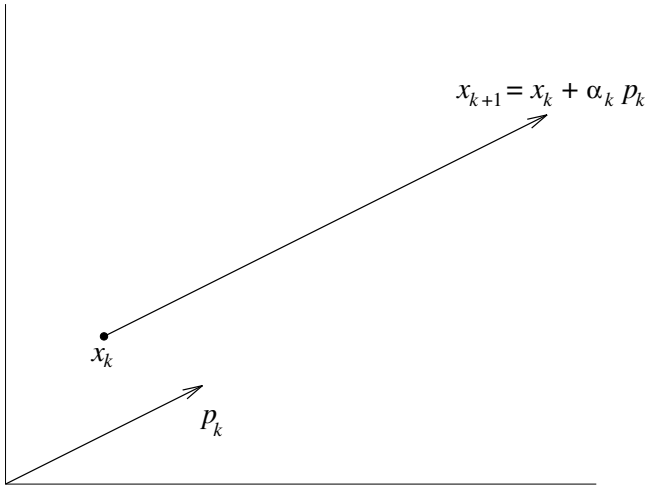

Figure 2.7. *General optimization algorithm.*

Why do we not just solve for the solution directly? Except for the simplest optimization problems, formulas for the solution do not exist. For example, consider the problem

$$\text{minimize } f(x) = e^x + x^2.$$

The optimality condition $f'(x) = 0$ has the form

$$e^x + 2x = 0,$$

but there is no simple formula for the solution to this equation. Hence for many problems some form of iterative method must be employed to determine a solution. (Any finite sequence of calculations is a formula of some sort, and so the solution of a general optimization problem can only be found as the limit of an infinite sequence. When we refer to computing a "solution" we most always mean an approximate solution, an element of this sequence that has sufficient accuracy. Determining the exact solution, or the limit of such a sequence, would be an "infinite" calculation.)

Why do we split the computation of x_{k+1} into two calculations? Ideally we would like to have $x_{k+1} = x_k + p_k$ where p_k solves

$$\underset{p}{\text{minimize}}\ f(x_k + p),$$

but this is equivalent to our original problem

$$\underset{x}{\text{minimize}}\ f(x).$$

Instead a compromise is employed. For an unconstrained problem of the form here, we will typically require that the search direction p_k be a *descent direction* for the function f at the point x_k. This means that for "small" steps taken along p_k the function value is guaranteed to decrease:

$$f(x_k + \alpha p_k) < f(x_k) \quad \text{for } 0 < \alpha \leq \epsilon$$

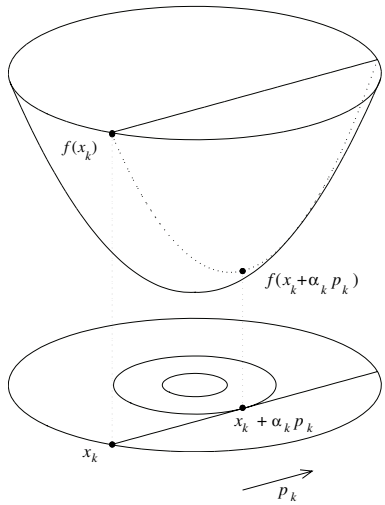

Figure 2.8. *Line search.*

for some ϵ. For a linear function $f(x) = c^T x$, p_k is a descent direction if

$$c^T(x_k + \epsilon p_k) = c^T x_k + \epsilon c^T p_k < c^T x_k,$$

or in other words if $c^T p_k < 0$. Techniques for computing descent directions for nonlinear functions are discussed in Chapter 11.

With p_k available, we would ideally like to determine the step length α_k so as to minimize the function in that direction:

$$\underset{\alpha \geq 0}{\text{minimize}} \; f(x_k + \alpha p_k).$$

This is a problem only involving one variable, the parameter α. The restriction $\alpha \geq 0$ is imposed because p_k is a descent direction.

Even for this one-dimensional problem there may not be a simple formula for the solution, so it too cannot normally be solved exactly. Instead, an α_k is computed that either "sufficiently decreases" the value of f or yields an "approximate minimizer" of the function f in the direction p_k. Both these terms have precise theoretical meanings that will be specified in later chapters, and computational techniques are available that allow α_k to be determined at reasonable cost. The calculation of α_k is called a *line search* because it corresponds to a search along the line $x_k + \alpha p_k$ defined by α. The line search is illustrated in Figure 2.8.

Algorithm II with its three major steps (the optimality test, computation of p_k, and computation of α_k) has been the basis for a great many of the most successful optimization algorithms ever developed. It has been used to develop many software packages for nonlinear optimization, and it is also present implicitly as part of the simplex method for linear programming. It is not the only approach possible (see Section 11.6), but it is the approach that we will emphasize in this book.

Using the concept of descent directions, we can establish an important condition for optimality for the constrained problem

$$\underset{x \in S}{\text{minimize}} \; f(x).$$

We define p to be a *feasible descent direction* at a point $x_k \in S$ if, for some $\epsilon > 0$,

$$x_k + \alpha p \in S \quad \text{and} \quad f(x_k + \alpha p) < f(x_k)$$

for all $0 < \alpha \leq \epsilon$. If a feasible descent direction exists at a point x_k, then it is possible to move a short distance along this direction to a feasible point with a better objective value. Then x_k cannot be a local minimizer for this problem. Hence, if x_* is a local minimizer, there cannot exist any feasible descent directions at x_*. This result will be used to derive optimality conditions for a variety of optimization problems.

Exercises

4.1. Let $x_k = (2, 1)^T$ and $p_k = (-1, 3)^T$. Plot the set $\{ x : x = x_k + \alpha p_k, \alpha \geq 0 \}$.

4.2. Find all descent directions for the linear function $f(x) = x_1 - 2x_2 + 3x_3$. Does your answer depend on the value of x?

4.3. Consider the problem

$$\begin{array}{ll} \text{minimize} & f(x) = -x_1 - x_2 \\ \text{subject to} & x_1 + x_2 \leq 2 \\ & x_1, x_2 \geq 0. \end{array}$$

(i) Determine the feasible directions at $x = (0, 0)^T$, $(0, 1)^T$, $(1, 1)^T$, and $(0, 2)^T$.

(ii) Determine whether there exist feasible descent directions at these points, and hence determine which (if any) of the points can be local minimizers.

2.5 Rates of Convergence

Many of the algorithms discussed in this book do not find a solution in a finite number of steps. Instead these algorithms compute a sequence of approximate solutions that we hope get closer and closer to a solution. When discussing such an algorithm, the following two questions are often asked:

- Does it converge?
- How fast does it converge?

It is the second question that is the topic of this section.

If an algorithm converges in a finite number of steps, the cost of that algorithm is often measured by counting the number of steps required, or by counting the number of arithmetic operations required. For example, if Gaussian elimination is applied to a system