1.8 NONDERIVATIVE METHODS

All the gradient methods examined so far require calculation of at least the gradient $\nabla f(x^k)$ and possibly the Hessian matrix $\nabla^2 f(x^k)$ at each generated point x^k. In many problems, either these derivatives are not available in explicit form or they are given by very complicated expressions. In such cases, it may be preferable to use the same algorithms as earlier with all unavailable derivatives approximated by finite differences.

First derivatives may be approximated by the *forward difference formula*

$$\frac{\partial f(x^k)}{\partial x^i} \approx \frac{1}{h}\big(f(x^k + he_i) - f(x^k)\big) \tag{8.1}$$

or by the *central difference formula*

$$\frac{\partial f(x^k)}{\partial x^i} \approx \frac{1}{2h}\big(f(x^k + he_i) - f(x^k - he_i)\big). \tag{8.2}$$

In these relations, h is a small positive scalar and e_i is the ith unit vector (ith column of the identity matrix). In some cases the same value of h can be used for all partial derivatives, but in other cases, particularly when the problem is poorly scaled, it is essential to use a different value of h for each partial derivative. This is a tricky process that often requires trial and error; see the following discussion.

The central difference formula requires twice as much computation as the forward difference formula. However, it is much more accurate. This can be seen by forming the corresponding Taylor series expansions, and by verifying that (in exact arithmetic) the absolute value of the error between the approximation and the actual derivatives is $O(h)$ for the forward difference formula, while it is $O(h^2)$ for the central difference formula. Note that if the central difference formula is used, one obtains at essentially no extra cost an approximation of each diagonal element of the Hessian using the formula

$$\frac{\partial^2 f(x^k)}{\big(\partial x_i\big)^2} \approx \frac{1}{h^2}\big(f(x^k + he_i) + f(x^k - he_i) - 2f(x^k)\big).$$

These approximations can be used in schemes based on diagonal scaling.

To reduce the approximation error, we would like to choose the finite difference interval h as small as possible. Unfortunately, there is a limit on how much h can be reduced due to the roundoff error that occurs when quantities of similar magnitude are subtracted by the computer. In particular, an error δ due to finite precision arithmetic in evaluating the numerator in Eq. (8.1) [or Eq. (8.2)], results in an error of δ/h (or $\delta/2h$, respectively) in the first derivative evaluation. Roundoff error is particularly evident in the approximate formulas (8.1) and (8.2) near a stationary point where ∇f is nearly zero, and the relative error size in the gradient approximation becomes very large.

Practical experience suggests that a good policy is to keep the scalar h for each derivative at a fixed value, which roughly balances the approximation error against the roundoff error. Based on the preceding calculations, this leads to the guideline

$$\frac{\delta}{h} = O(h) \quad \text{or} \quad h = O(\delta^{1/2}), \quad \text{for the forward difference formula (8.1),}$$

$$\frac{\delta}{2h} = O(h^2) \quad \text{or} \quad h = O(\delta^{1/3}), \quad \text{for the central difference formula (8.2),}$$

where δ is the error due to finite precision arithmetic in evaluating the numerator in Eq. (8.1) [or Eq. (8.2)]. Thus, a much larger value of h can be used in conjunction with the central difference formula. A good practical rule is to use the forward formula (8.1) until the absolute value of the corresponding approximate derivative becomes less than a certain tolerance; i.e.,

$$\left| \frac{f(x^k + he_i) - f(x^k)}{h} \right| \leq \epsilon,$$

where $\epsilon > 0$ is some prespecified scalar. At that point, a switch to the central difference formula should be made.

Second derivatives may be approximated by the forward difference formula

$$\frac{\partial^2 f(x^k)}{\partial x_i \partial x_j} \approx \frac{1}{h} \left(\frac{\partial f(x^k + he_j)}{\partial x^i} - \frac{\partial f(x^k)}{\partial x_i} \right) \tag{8.3}$$

or the central difference formula

$$\frac{\partial^2 f(x^k)}{\partial x_i \partial x_j} \approx \frac{1}{2h} \left(\frac{\partial f(x^k + he_j)}{\partial x_i} - \frac{\partial f(x^k - he_j)}{\partial x_i} \right). \tag{8.4}$$

Practical experience suggests that in discretized forms of Newton's method, extreme accuracy in approximating second derivatives is not very important in terms of rate of convergence. For this reason, exclusive use of the forward difference formula (8.3) is adequate in most cases. However, one should certainly check for positive definiteness of the discretized Hessian approximation and introduce modifications if necessary, as discussed in Section 1.4.

1.8.1 Coordinate Descent

There are several other nonderivative methods for minimizing differentiable functions. A particularly important algorithm is the *coordinate descent method*. Here the cost is minimized along one coordinate direction at each iteration. The order in which coordinates are chosen may vary in the course of the algorithm. In the case where this order is cyclical, given x^k, the ith coordinate of x^{k+1} is determined by

$$x_i^{k+1} = \arg\min_{\xi \in \Re} f(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, \xi, x_{i+1}^k, \ldots, x_n^k); \qquad (8.5)$$

see Fig. 1.8.1. The method can also be used for minimization of f subject to upper and lower bounds on the variables x^i (the minimization over $\xi \in \Re$ in the preceding equation is replaced by minimization over the appropriate interval). We will analyze the method within this more general context in the next chapter.

An important advantage of the coordinate descent method is that it is well suited for *parallel computation*. In particular, suppose that there is a subset of coordinates $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$, which are not coupled through the cost function, that is, $f(x)$ can be written as $\sum_{r=1}^m f_{i_r}(x)$, where for each r, $f_{i_r}(x)$ does not depend on the coordinates x_{i_s} for all $s \neq r$. Then one can perform the m coordinate descent iterations

$$x_{i_r}^{k+1} = \arg\min_{\xi} f_{i_r}(x^k + \xi e_{i_r}), \qquad r = 1, \ldots, m,$$

independently and in parallel. Thus, in problems with special structure where the set of coordinates can be partitioned into p subsets with the independence property just described, one can perform a full cycle of coordinate descent iterations in p (as opposed to n) parallel steps (assuming of course that a sufficient number of parallel processors is available).

The coordinate descent method generally has similar convergence properties to steepest descent. For continuously differentiable cost functions, it can be shown to generate sequences whose limit points are stationary, although the proof of this is sometimes complicated and requires some additional assumptions (see Prop. 2.7.1 in Section 2.7, which deals with a constrained version of coordinate descent and requires strict convexity of the cost function along each coordinate). There is also a great deal of analysis of coordinate descent in a context where its use is particularly favorable, namely in solving dual problems (see Section 6.2). Within this context, the strict convexity assumption is neither satisfied nor is it essential (see the references given in Chapter 6). The convergence rate of coordinate descent to nonsingular and singular local minima can be shown to be linear and sublinear, respectively, similar to steepest descent. Often, the choice between coordinate descent and steepest descent is dictated by the structure of the cost function. Both methods can be very slow, but for many practical contexts, they can be quite effective.

1.8.2 Direct Search Methods

In the coordinate descent method we search along the fixed set of coordinate directions and we are guaranteed a cost improvement at a nonstationary point because these directions are linearly independent. This

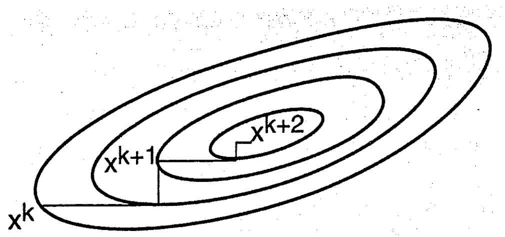

Figure 1.8.1. Illustration of the coordinate descent method.

idea can be generalized by using a different set of directions and by occasionally changing this set of directions with the aim of accelerating convergence. There are a number of methods of this type: Rosenbrock's method [Ros60a], the pattern search algorithm of Hooke and Jeeves [HoJ61], and the simplex algorithms of Spendley, Hext, and Himsworth [SHH62], and Nelder and Mead [NeM65]. Unfortunately, the rationale of these methods often borders on the heuristic, and their theoretical convergence properties are often unsatisfactory. However, these methods are often fairly simple to implement and like the coordinate descent method, they do not require gradient calculations. We describe the Nelder and Mead simplex method (not to be confused with the simplex method of linear programming), which has enjoyed considerable popularity.

At the typical iteration of this method, we start with a *simplex*, that is, the convex hull of $n+1$ points, $x^0, x^1, \ldots, x^n$, and we end up with another simplex. Let x_{min} and x_{max} denote the "best" and "worst" vertices of the simplex, that is the vertices satisfying

$$f(x_{min}) = \min_{i=0,1,\ldots,n} f(x^i), \tag{8.6}$$

$$f(x_{max}) = \max_{i=0,1,\ldots,n} f(x^i). \tag{8.7}$$

Let also $\hat{x}$ denote the centroid of the face of the simplex formed by the vertices other than x_{max}

$$\hat{x} = \frac{1}{n}\left(-x_{max} + \sum_{i=0}^{n} x^i\right). \tag{8.8}$$

The iteration replaces the worst vertex x_{max} by a "better" one. In particular, the *reflection point* $x_{ref} = 2\hat{x} - x_{max}$ is computed, which lies on the line passing through x_{max} and $\hat{x}$, and is symmetric to x_{max} with respect to $\hat{x}$. Depending on the cost value of x_{ref} relative to the points of the simplex other than x_{max}, a new vertex x_{new} is computed, and a new simplex is formed from the old by replacing the vertex x_{max} by x_{new}, while keeping the other n vertices.

Typical Iteration of the Simplex Method

Step 1: (Reflection Step) Compute

$$x_{ref} = 2\hat{x} - x_{max}. \tag{8.9}$$

Then compute x_{new} according to the following three cases:

(1) (**x_{ref} has min cost**) If $f(x_{min}) > f(x_{ref})$, go to Step 2.

(2) (**x_{ref} has intermediate cost**) If $\max\{f(x^i) \mid x^i \neq x_{max}\} > f(x_{ref}) \geq f(x_{min})$, go to Step 3.

(3) (**x_{ref} has max cost**) If $f(x_{ref}) \geq \max\{f(x^i) \mid x^i \neq x_{max}\}$, go to Step 4.

Step 2: (Attempt Expansion) Compute

$$x_{exp} = 2x_{ref} - \hat{x}. \tag{8.10}$$

Define

$$x_{new} = \begin{cases} x_{exp} & \text{if } f(x_{exp}) < f(x_{ref}), \\ x_{ref} & \text{otherwise,} \end{cases}$$

and form the new simplex by replacing the vertex x_{max} with x_{new}.

Step 3: (Use Reflection) Define $x_{new} = x_{ref}$, and form the new simplex by replacing the vertex x_{max} with x_{new}.

Step 4: (Perform Contraction) Define

$$x_{new} = \begin{cases} \frac{1}{2}(x_{max} + \hat{x}) & \text{if } f(x_{max}) \leq f(x_{ref}), \\ \frac{1}{2}(x_{ref} + \hat{x}) & \text{otherwise,} \end{cases} \tag{8.11}$$

and form the new simplex by replacing the vertex x_{max} with x_{new}.

The reflection step and the subsequent possible steps of the iteration are illustrated in Fig. 1.8.2 (a)-(d). The entire method is illustrated in Fig. 1.8.3. Exercise 8.3 shows a cost improvement property of the method in the case where f is strictly convex. However, there are no known convergence results for the method. Furthermore, when the cost function is not convex, it is possible that the new simplex vertex x_{new} has larger cost value than the old vertex x_{max}. In this case a modification that has been suggested is to "shrink" the old simplex towards the best vertex x_{min}, that is, form a new simplex by replacing all the vertices x^i, $i = 0, 1, \ldots, n$, by

$$\overline{x}^i = \tfrac{1}{2}(x^i + x_{min}), \qquad i = 0, 1, \ldots, n.$$

The method as given above seems to work reasonably well in practice, particularly for problems of relatively small dimension (say up to 10). However, it is not guaranteed to have desirable convergence properties, and in fact a convergence counterexample is given in [McK94]. Reference

[Tse95a] provides a relatively simple modification with satisfactory convergence properties. There are also a number of related methods, some of which have demonstrable convergence properties; see [DeT91] and [Tor91]. Note that the constants used in Eqs. (8.9), (8.10), and (8.11) are somewhat arbitrary, as suggested by the interpretation of the method given in Figs. 1.8.2 and 1.8.3. More general forms of these equations are

$$x_{ref} = \hat{x} + \beta(\hat{x} - x_{max}),$$

$$x_{exp} = x_{ref} + \gamma(x_{ref} - \hat{x}),$$

$$x_{con} = \begin{cases} \theta x_{max} + (1-\theta)\hat{x} & \text{if } f(x_{max}) \le f(x_{ref}), \\ \theta x_{ref} + (1-\theta)\hat{x} & \text{otherwise,} \end{cases}$$

where $\beta > 0$, $\gamma > 0$, and $\theta \in (0,1)$ are scalars known as the *reflection coefficient*, the *expansion coefficient*, and the *contraction coefficient*, respectively. The formulas of Eqs. (8.9)-(8.11) correspond to $\beta = 1$, $\gamma = 1$, and $\theta = 1/2$, respectively.

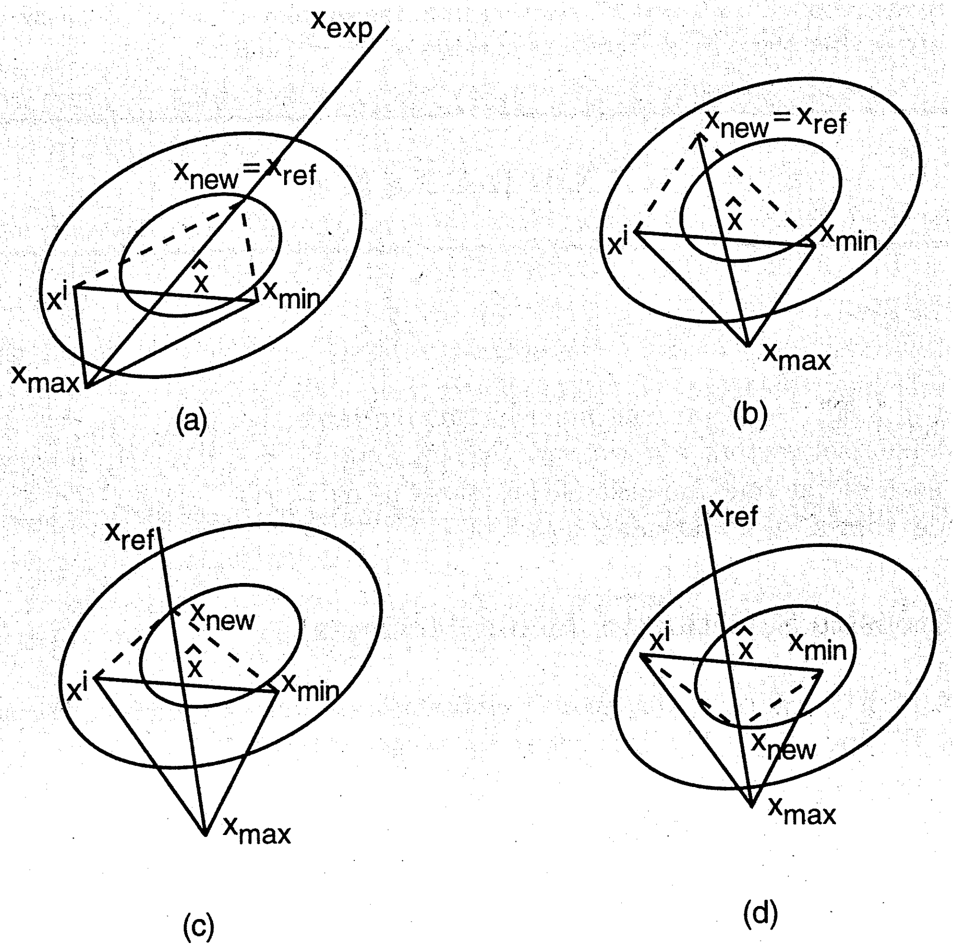

Figure 1.8.2. Illustration of the reflection step and the possible subsequent steps of an iteration of the simplex method. In (a), the new vertex x_{new} is determined via the expansion Step 2. In (b), x_{new} is determined via Step 3, and the reflection step is accepted. In (c) and (d), x_{new} is determined via the contraction Step 4.

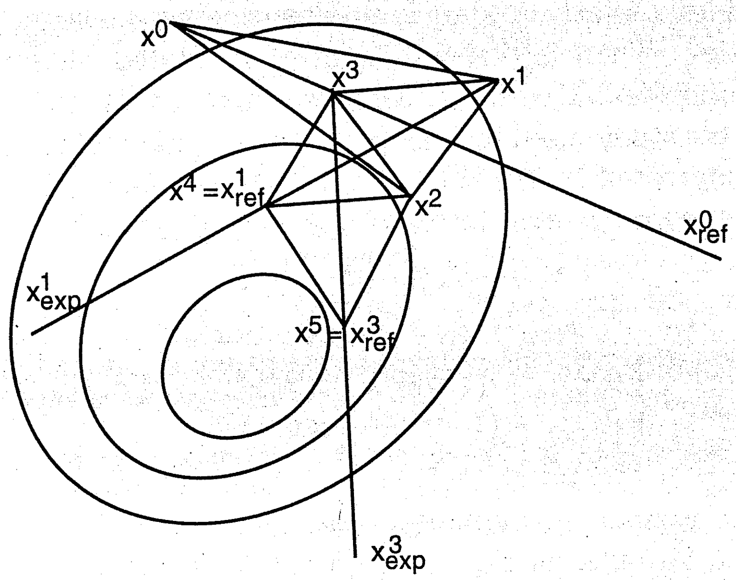

Figure 1.8.3. Illustration of three iterations of the simplex method, which generate the points x^3, x^4, and x^5, starting from the simplex x^0, x^1, x^2. The simplex obtained after these three iterations consists of x^2, x^4, and x^5.

EXERCISES

8.1

Let $f : \Re^n \mapsto \Re$ be continuously differentiable, let $p_1, \ldots, p_n$ be linearly independent vectors, and suppose that for some x^*, $\alpha = 0$ is a stationary point of each of the one-dimensional functions $g_i(\alpha) = f(x^* + \alpha p_i)$, $i = 1, \ldots, n$. Show that x^* is a stationary point of f.

8.2 (Stepsize Selection in Jacobi Methods)

Let $f : \Re^n \mapsto \Re$ be a continuously differentiable convex function. For a given $x \in \Re^n$ and all $i = 1, \ldots, n$, define the vector $\overline{x}$ by

$$\overline{x}_i = \arg\min_{\xi \in \Re} f(x_1, \ldots, x_{i-1}, \xi, x_{i+1}, \ldots, x_n).$$

The Jacobi method is defined by the iteration

$$x := x + \alpha(\overline{x} - x),$$

where α is a positive stepsize parameter.

(a) Show that if x does not minimize f, then the Jacobi iteration reduces the value of f when $\alpha = 1/n$.