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 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 Section 3.4, and 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

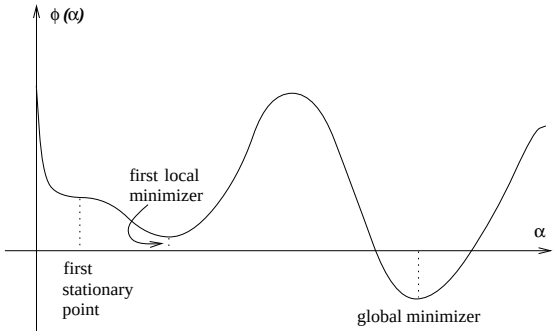

Figure 3.1: The ideal step length is the global minimizer.

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.

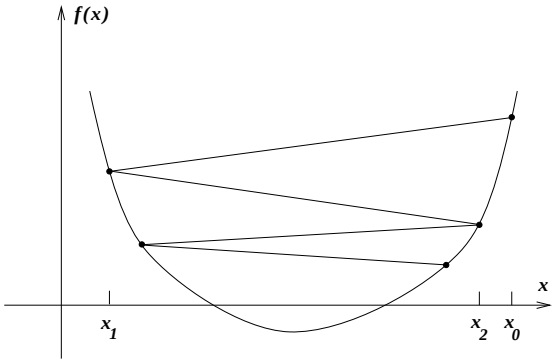

Figure 3.2: Insufficient reduction in f.

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 line search algorithms 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 to require a reduction in f, that is, $f(x_k + \alpha_k p_k) < f(x_k)$. That this requirement is not enough to produce convergence to x^* is illustrated in Figure 3.2, for which the minimum function value is $f^* = -1$, but a sequence of iterates $\{x_k\}$ for which $f(x_k) = 5/k$, $k = 0, 1, \ldots$ yields a decrease at each iteration but has a limiting function value of zero. The difficulty is that there is not a *sufficient* reduction in f at each step, 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:

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

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 $c_1 = 10^{-4}$.

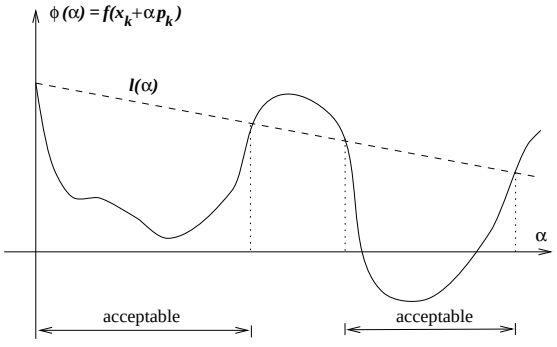

Figure 3.3: Sufficient decrease condition.

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 ϕ at α_k is greater than c_2 times the initial slope $\phi'(0)$. 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 $\phi'(\alpha_k)$ is only slightly negative or even positive, it is a sign that we cannot expect much more decrease in f in this direction, 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.

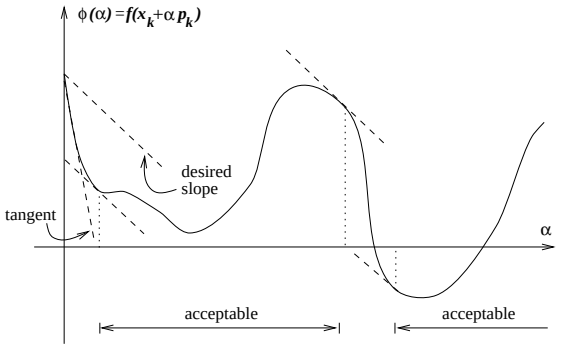

Figure 3.4: The curvature condition.

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:

$$\begin{aligned} f(x_k + \alpha_k p_k) &\leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, & \text{(3.6a)} \\ \nabla f(x_k + \alpha_k p_k)^T p_k &\geq c_2 \nabla f_k^T p_k, & \text{(3.6b)} \end{aligned}$$

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

$$\begin{aligned} f(x_k + \alpha_k p_k) &\leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, & \text{(3.7a)} \\ |\nabla f(x_k + \alpha_k p_k)^T p_k| &\leq c_2 |\nabla f_k^T p_k|, & \text{(3.7b)} \end{aligned}$$

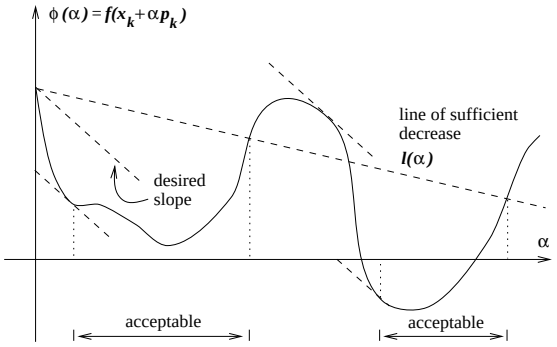

Figure 3.5: Step lengths satisfying the Wolfe conditions.

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 6.

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:

$$f(x_k) + (1-c)\alpha_k \nabla f_k^T p_k \leq f(x_k + \alpha_k p_k) \leq f(x_k) + c\alpha_k \nabla f_k^T p_k, \qquad (3.11)$$

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

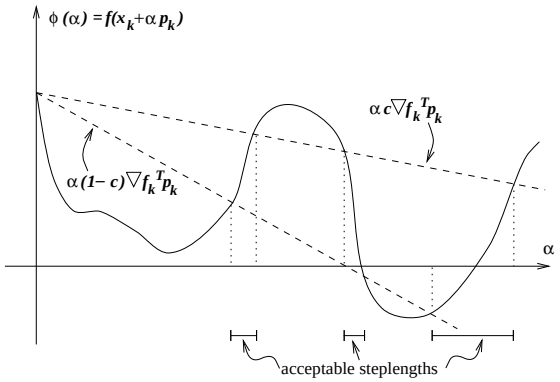

Figure 3.6: The Goldstein conditions.

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.

Algorithm 3.1 (Backtracking Line Search)
Choose $\bar{\alpha} > 0$, $\rho, c \in (0,1)$; set $\alpha \leftarrow \bar{\alpha}$;
repeat until $f(x_k + \alpha p_k) \leq f(x_k) + c\alpha \nabla f_k^T p_k$
$\alpha \leftarrow \rho\alpha$;
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 $\rho \in [\rho_{\rm lo}, \rho_{\rm hi}]$, for some fixed constants $0 < \rho_{\rm lo} < \rho_{\rm hi} < 1$.

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 a factor ρ 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 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 quantifies the effect of properly chosen step lengths α_k, and 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 $\mathcal{L} \stackrel{\text{def}}{=} \{x : f(x) \leq f(x_0)\}$, 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*

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

Then

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

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

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

while the Lipschitz condition (3.13) implies that

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

By combining these two relations, we obtain

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

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

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

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

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

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

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

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. For all these strategies the step length selection implies inequality (3.14), which we call the *Zoutendijk condition*.

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 7 and 6) and are often satisfied in practice.

The Zoutendijk condition (3.14), 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

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

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 is parallel to 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

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

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

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

(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

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

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

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

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.

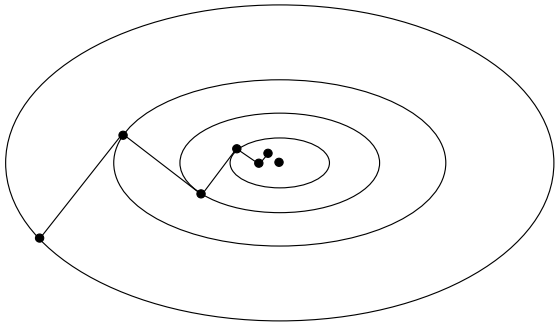

Figure 3.7: Steepest descent steps.

To quantify the rate of convergence we introduce the weighted norm $\|x\|_Q^2 = x^T Q x$. 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

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

(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*

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

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

The proof of this result is given by Luenberger [171]. 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. Let r be any scalar satisfying*

$$r \in \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}, 1 \right),$$

where $\lambda_1 \leq \cdots \leq \lambda_n$ are the eigenvalues of $\nabla^2 f(x^*)$. Then for all k sufficiently large, we have

$$f(x_{k+1}) - f(x^*) \leq r^2 [f(x_k) - f(x^*)].$$

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 6. 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 twice 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 $c_1 \leq \frac{1}{2}$. 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 twice 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

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

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

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

giving the superlinear convergence result. □

We will see in Chapter 6 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 Section 3.4 and Chapter 4 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.38)) 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, there is a radius $r > 0$ such that $\|\nabla^2 f_k^{-1}\| \le 2\|\nabla^2 f(x^*)^{-1}\|$ for all x_k with $\|x_k - x^*\| \le r$. 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 $\tilde{L} = L\|\nabla^2 f(x^*)^{-1}\|$. Choosing x_0 so that $\|x_0 - x^*\| \le \min(r, 1/(2\tilde{L}))$, we can use this inequality inductively to deduce that 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

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

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. [1]

3.4 Newton's Method with Hessian Modification

Away from the solution, the Hessian matrix $\nabla^2 f(x)$ may not be positive definite, and therefore, the Newton direction p_k^{N} defined by

$$\nabla^2 f(x_k) p_k^{\mathrm{N}} = -\nabla f(x_k) \tag{3.38}$$

may not be a descent direction. We now describe an approach to overcome this difficulty when a direct linear algebra technique, such as Gaussian elimination, is used to solve the Newton equations (3.38). It consists of modifying the Hessian matrix before or during the solution process so that the computed direction p_k satisfies a linear system identical to (3.38), except that the coefficient matrix is replaced with a positive definite approximation. The modified Hessian is obtained by adding either a positive diagonal matrix or a full matrix to the true Hessian $\nabla^2 f(x_k)$. Following is a general description of this method.

Algorithm 3.2 (Line Search Newton with Modification)
given initial point x_0;
for $k = 0, 1, 2, \ldots$

[1] Coordinate Descent Methods were moved to the DFO chapter

Factorize the matrix $B_k = \nabla^2 f(x_k) + E_k$, where $E_k = 0$ if $\nabla^2 f(x_k)$ is sufficiently positive definite; otherwise, E_k is chosen to ensure that B_k is sufficiently positive definite;
Solve $B_k p_k = -\nabla f(x_k)$;
Set $x_{k+1} = x_k + \alpha_k p_k$, where α_k satisfies the Wolfe, Goldstein, or Armijo backtracking conditions;

end

Some approaches do not compute E_k explicitly, but rather introduce extra steps and tests into standard factorization procedures, modifying these procedures "on the fly" so that the computed factors are in fact the factors of a positive definite matrix. Strategies based on modifying a Cholesky factorization and on modifying a symmetric indefinite factorization of the Hessian are described in the next section.

Algorithm 3.2 is a practical Newton method that can be applied from any starting point. We can establish fairly satisfactory global convergence results for it, provided that the strategy for choosing E_k (and hence B_k) satisfies the *bounded modified factorization* property. This property is that the matrices in the sequence $\{B_k\}$ have bounded condition number whenever the sequence of Hessians $\{\nabla^2 f(x_k)\}$ is bounded; that is,

$$\text{cond}(B_k) = \|B_k\| \, \|B_k^{-1}\| \leq C, \quad \text{for some } C > 0 \text{ and all } k = 0, 1, 2, \ldots. \tag{3.39}$$

If this property holds, global convergence of the modified line search Newton method follows directly from the results of Section 3.2, as we show in the following result.

Theorem 3.8 *Let f be twice continuously differentiable on an open set $\mathcal{D}$, and assume that the starting point x_0 of Algorithm 3.2 is such that the level set $L = \{x \in \mathcal{D} : f(x) \leq f(x_0)\}$ is compact. Then if the bounded modified factorization property holds, we have that*

$$\lim_{k \to \infty} \nabla f(x_k) = 0.$$

Proof. The line search ensures that all iterates x_k remain in the level set L. Since $\nabla^2 f(x)$ is assumed to be continuous on $\mathcal{D}$, and since L is compact, we have that the sequence of Hessians $\{\nabla^2 f(x_k)\}$ is bounded, and therefore (3.39) holds. Since Algorithm 3.2 uses one of the line searches for which Zoutendijk's theorem (Theorem 3.2) holds, the result follows from (3.16). □

We now consider the convergence rate of Algorithm 3.2. Suppose that the sequence of iterates x_k converges to a point x^* where $\nabla^2 f(x^*)$ is sufficiently positive definite in the sense that the modification strategies described in the next section return the modification $E_k = 0$ for all sufficiently large k. By Theorem 3.5, we have that $\alpha_k = 1$ for all sufficiently large k, so that Algorithm 3.2 reduces to a pure Newton method, and the rate of convergence is quadratic.

For problems in which ∇f^* is close to singular, there is no guarantee that the modification E_k will eventually vanish, and the convergence rate may be only linear. Besides requiring the modified matrix B_k to be well conditioned (so that Theorem 3.8 holds), we would like the modification to be as small as possible, so that the second-order information in the Hessian is preserved as far as possible. Naturally, we would also like the modified factorization to be computable at moderate cost.

To set the stage for the matrix factorization techniques that will be used in Algorithm 3.2 we will begin by assuming that the eigenvalue decomposition of $\nabla^2 f(x_k)$ is available. This is not realistic for large-scale problems because this decomposition is generally too expensive to compute, but it will motivate several practical modification strategies.

Eigenvalue Modification

Consider a problem in which, at the current iterate x_k, $\nabla f(x_k) = (1, -3, 2)$ and $\nabla^2 f(x_k) = \text{diag}(10, 3, -1)$, which is clearly indefinite. By the spectral decomposition theorem (see the Appendix) we can define $Q = I$ and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \lambda_3)$, and write

$$\nabla^2 f(x_k) = Q\Lambda Q^T = \sum_{i=1}^{n} \lambda_i q_i q_i^T. \tag{3.40}$$

The pure Newton step—the solution of (3.38)—is $p_k^{\text{N}} = (-0.1, 1, 2)$, which is not a descent direction, since $\nabla f(x_k)^T p_k^{\text{N}} > 0$. One might suggest a modified strategy in which we replace $\nabla^2 f(x_k)$ by a positive definite approximation B_k, in which all negative eigenvalues in $\nabla^2 f(x_k)$ are replaced by a small positive number δ that is somewhat larger than machine precision $\mathbf{u}$; say $\delta = \sqrt{\mathbf{u}}$. Assuming that machine precision is 10^{-16}, the resulting matrix in our example is

$$B_k = \sum_{i=1}^{2} \lambda_i q_i q_i^T + \delta q_3 q_3^T = \text{diag}\left(10, 3, 10^{-8}\right), \tag{3.41}$$

which is numerically positive definite and whose curvature along the eigenvectors q_1 and q_2 has been preserved. Note, however, that the search direction based on this modified Hessian is

$$p_k = -B_k^{-1}\nabla f_k = -\sum_{i=1}^{2} \frac{1}{\lambda_i} q_i \big(q_i^T \nabla f_k\big) - \frac{1}{\delta} q_3 \big(q_3^T \nabla f(x_k)\big) \approx -\big(2 \times 10^8\big) q_3. \tag{3.42}$$

For small δ, this step is nearly parallel to q_3 (with relatively small contributions from q_1 and q_2) and very long. Although f decreases along the direction p_k, its extreme length violates the spirit of Newton's method, which relies on a quadratic approximation of the objective function that is valid in a neighborhood

of the current iterate x_k. It is therefore questionable that this search direction is effective.

A different type of search direction would be obtained by flipping the signs of the negative eigenvalues in (3.40), which amounts to setting $\delta = 1$ in our example. Again, there is no consensus as to whether this constitutes a desirable modification to Newton's method. Various other strategies can be considered: We could set the last term in (3.42) to zero, so that the search direction has no components along the negative curvature directions, or we could adapt the choice of δ to ensure that the length of the step is not excessive. (This last strategy has the flavor of trust-region methods.) We see that there is a great deal of freedom in devising modification strategies, and there is currently no agreement on which is the ideal strategy.

Setting the issue of the choice of δ aside for the moment, let us look more closely at the process of modifying a matrix so that it becomes positive definite. The modification (3.41) to the example matrix (3.40) can be shown to be optimal in the following sense. If A is a symmetric matrix with spectral decomposition $A = Q\Lambda Q^T$, then the correction matrix ΔA of *minimum Frobenius norm* that ensures that $\lambda_{\min}(A + \Delta A) \geq \delta$ is given by

$$\Delta A = Q \text{ diag } (\tau_i) Q^T, \qquad \text{with} \qquad \tau_i = \begin{cases} 0, & \lambda_i \geq \delta, \\ \delta - \lambda_i, & \lambda_i < \delta. \end{cases} \tag{3.43}$$

Here $\lambda_{\min}(A)$ denotes the smallest eigenvalue of A, and the Frobenius norm of a matrix is defined as $\|A\|_F^2 = \sum_{i,j=1}^n a_{ij}^2$ (see (A.9)). Note that ΔA is not diagonal in general, and that the modified matrix is given by

$$A + \Delta A = Q(\Lambda + \text{diag}(\tau_i))Q^T.$$

By using a different norm we can obtain a *diagonal* modification. Suppose again that A is a symmetric matrix with spectral decomposition $A = Q\Lambda Q^T$. A correction matrix ΔA with minimum Euclidean norm that satisfies $\lambda_{\min}(A + \Delta A) \geq \delta$ is given by

$$\Delta A = \tau I, \qquad \text{with} \qquad \tau = \max(0, \delta - \lambda_{\min}(A)). \tag{3.44}$$

The modified matrix now has the form

$$A + \tau I, \tag{3.45}$$

which happens to have the same form as the matrix occurring in (unscaled) trust–region methods (see Chapter 4). All the eigenvalues of (3.45) have thus been shifted, and all are greater than δ.

These results suggest that both diagonal and nondiagonal modifications can be considered. Even though we have not answered the question of what constitutes a good modification, various practical diagonal and nondiagonal modifications have been proposed and implemented in software. They do not make use of the spectral decomposition of the Hessian, since this is generally too expensive to compute. Instead, they use Gaussian elimination, choosing the modifications

indirectly and hoping that somehow they will produce good steps. Numerical experience indicates that the strategies described next often (but not always) produce good search directions.

Adding a Multiple of the Identity

Perhaps the simplest idea is to find a scalar $\tau > 0$ such that $\nabla^2 f(x_k) + \tau I$ is sufficiently positive definite. From the previous discussion we know that τ must satisfy (3.44), but we normally don't have a good estimate of the smallest eigenvalue of the Hessian. A simple idea is to try successively larger values of τ, as described in the following algorithm. Here a_{ii} denotes a diagonal element of A.

Algorithm 3.3 (Cholesky with Added Multiple of the Identity)
Choose $\beta > 0$;
if $\min_i a_{ii} > 0$, set $\tau_0 \leftarrow 0$;
else $\tau_0 = -\min(a_{ii}) + \beta$;
end
for $k = 0, 1, 2, \ldots$
Attempt to apply the Cholesky algorithm to obtain $LL^T = A + \tau_k I$;
if the factorization is completed successfully **stop** and return L;
else $\tau_{k+1} \leftarrow \max(2\tau_k, \beta)$;
end (if)
end (for)

The choice of β is heuristic; a typical value is $\beta = 10^{-3}$. We could choose the first nonzero shift τ_0 to be proportional to be the final value of τ used in the latest Hessian modification; see also Algorithm A.4. This strategy implemented in Algorithm 3.3 is quite simple and may be preferable to the modified factorization techniques described next, but it suffers from one drawback. Every value of τ_k requires a new factorization of $A + \tau_k I$, which can be quite expensive if several trial values are generated. Therefore it may be advantageous to increase τ more rapidly, say by a factor of 10 instead of 2 in the last else-statement.

Modified Cholesky Factorization

Another approach for modifying a Hessian matrix that is not positive definite is to perform a Cholesky factorization of $\nabla^2 f(x_k)$, but to increase the diagonal elements encountered during the factorization (where necessary) to ensure that they are sufficiently positive. This *modified Cholesky* approach is designed to accomplish two goals: It guarantees that the modified Cholesky factors exist and are bounded relative to the norm of the actual Hessian, and it does not modify the Hessian if it is sufficiently positive definite.

We begin our description of this approach by briefly reviewing the Cholesky factorization. Every symmetric and positive definite matrix A can be written as

$$A = LDL^T, \tag{3.46}$$

where L is a lower triangular matrix with unit diagonal elements and D is a diagonal matrix with positive elements on the diagonal. By equating the elements in (3.46), column by column, it is easy to derive formulas for computing L and D.

Example 3.1
Consider the case $n = 3$. The equation $A = LDL^T$ is given by

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} \begin{bmatrix} 1 & l_{21} & l_{31} \\ 0 & 1 & l_{32} \\ 0 & 0 & 1 \end{bmatrix}.$$

(The notation indicates that A is symmetric.) By equating the elements of the first column, we have

$$\begin{array}{lcll} a_{11} & = & d_1, & \\ a_{21} & = & d_1 l_{21} & \Rightarrow \quad l_{21} = a_{21}/d_1, \\ a_{31} & = & d_1 l_{31} & \Rightarrow \quad l_{31} = a_{31}/d_1. \end{array}$$

Proceeding with the next two columns, we obtain

$$\begin{array}{lcll} a_{22} & = & d_1 l_{21}^2 + d_2 & \Rightarrow \quad d_2 = a_{22} - d_1 l_{21}^2, \\ a_{32} & = & d_1 l_{31} l_{21} + d_2 l_{32} & \Rightarrow \quad l_{32} = \left(a_{32} - d_1 l_{31} l_{21}\right)/d_2, \\ a_{33} & = & d_1 l_{31}^2 + d_2 l_{32}^2 + d_3 & \Rightarrow \quad d_3 = a_{33} - d_1 l_{31}^2 - d_2 l_{32}^2. \end{array}$$

This procedure is generalized in the following algorithm.

Algorithm 3.4 (Cholesky Factorization, LDL^T Form)
for $j = 1, 2, \ldots, n$
  $c_{jj} \leftarrow a_{jj} - \sum_{s=1}^{j-1} d_s l_{js}^2$;
  $d_j \leftarrow c_{jj}$;
  for $i = j+1, \ldots, n$
    $c_{ij} \leftarrow a_{ij} - \sum_{s=1}^{j-1} d_s l_{is} l_{js}$;
    $l_{ij} \leftarrow c_{ij}/d_j$;
  end
end

One can show (see, for example, Golub and Van Loan [117, Section 4.2.3]) that the diagonal elements d_{jj} are all positive whenever A is positive definite. The scalars c_{ij} have been introduced only to facilitate the description of the modified factorization discussed below. We should note that Algorithm 3.4 differs a little from the standard form of the Cholesky factorization, which produces a lower triangular matrix M such that

$$A = MM^T. \tag{3.47}$$

In fact, we can make the identification $M = LD^{1/2}$ to relate M to the factors L and D computed in Algorithm 3.4. The technique for computing M appears as Algorithm A.2 in the Appendix.

If A is indefinite, the factorization $A = LDL^T$ may not exist. Even if it does exist, Algorithm 3.4 is numerically unstable when applied to such matrices, in the sense that the elements of L and D can become arbitrarily large. It follows that a strategy of computing the LDL^T factorization and then modifying the diagonal after the fact to force its elements to be positive may break down, or may result in a matrix that is drastically different from A.

Instead, we can modify the matrix A during the course of the factorization in such a way that all elements in D are sufficiently positive, and so that the elements of D and L are not too large. To control the quality of the modification, we choose two positive parameters δ and β and require that during the computation of the jth columns of L and D in Algorithm 3.4 (that is, for each j in the outer loop of the algorithm) the following bounds be satisfied:

$$d_j \geq \delta, \qquad |m_{ij}| \leq \beta, \quad i = j+1, \ldots, n. \tag{3.48}$$

To satisfy these bounds we only need to change one step in Algorithm 3.4: The formula for computing the diagonal element d_j in Algorithm 3.4 is replaced by

$$d_j = \max\left(|c_{jj}|, \left(\frac{\theta_j}{\beta}\right)^2, \delta\right), \quad \text{with } \theta_j = \max_{j<i\leq n} |c_{ij}|. \tag{3.49}$$

To verify that (3.48) holds, we note from Algorithm 3.4 that $c_{ij} = l_{ij}d_j$, and therefore

$$|m_{ij}| = |l_{ij}\sqrt{d_j}| = \frac{|c_{ij}|}{\sqrt{d_j}} \leq \frac{|c_{ij}|\beta}{\theta_j} \leq \beta, \quad \text{for all } i > j.$$

We note that θ_j can be computed prior to d_j because the elements c_{ij} in the second **for** loop of Algorithm 3.4 do not involve d_j. In fact, this is the reason for introducing the quantities c_{ij} into the algorithm.

These observations are the basis of the modified Cholesky algorithm described in detail in Gill, Murray, and Wright [107], which introduces symmetric interchanges of rows and columns to try to reduce the size of the modification. If P denotes the permutation matrix associated with the row and column interchanges, the algorithm produces the Cholesky factorization of the permuted, modified matrix $PAP^T + E$, that is,

$$PAP^T + E = LDL^T = MM^T, \tag{3.50}$$

where E is a nonnegative diagonal matrix that is zero if A is sufficiently positive definite. One can show (Moré and Sorensen [192]) that the matrices B_k obtained by this modified Cholesky algorithm to the exact Hessians $\nabla^2 f(x_k)$ have bounded condition numbers, that is, the bound (3.39) holds for some value of C.

Gershgorin Modification

We now give a brief outline of an alternative modified Cholesky algorithm that makes use of Gershgorin circle estimates to increase the diagonal elements as necessary. The first step of this strategy is to apply Algorithm **??** to the matrix A, terminating with the usual factorization (3.50). If the perturbation matrix E is zero, we are finished, since in this case A is already sufficiently positive definite. Otherwise, we compute two upper bounds b_1 and b_2 on the smallest eigenvalue $\lambda_{\min}(A)$ of A. The first estimate b_1 is obtained from the Gershgorin circle theorem; it guarantees that $A + b_1 I$ is strictly diagonally dominant. The second upper bound b_2 is simply

$$b_2 = \max_{1 \le i \le n} e_{ii}.$$

We now define

$$\mu = \min(b_1, b_2),$$

and conclude the algorithm by computing the factorization of $A + \mu I$, taking the Cholesky factor of this matrix as our modified Cholesky factor. The use of b_2 gives a much needed control on the process, since the estimate b_1 obtained from the Gershgorin circle theorem can be quite loose.

It is not known whether this alternative is preferable to Algorithm **??** alone. Neither strategy will modify a sufficiently positive definite matrix A, but it is difficult to quantify the meaning of the term "sufficient" in terms of the smallest eigenvalue $\lambda_{\min}(A)$. Both strategies may in fact modify a matrix A whose minimum eigenvalue is greater than the parameter δ of Algorithm **??**.

Modified Symmetric Indefinite Factorization

Another strategy for modifying an indefinite Hessian is to use a procedure based on a symmetric indefinite factorization. Any symmetric matrix A, whether positive definite or not, can be written as

$$PAP^T = LBL^T, \tag{3.51}$$

where L is unit lower triangular, B is a block diagonal matrix with blocks of dimension 1 or 2, and P is a permutation matrix (see Golub and Van Loan [117, Section 4.4]). We mentioned earlier that attempting to compute the LDL^T factorization of an indefinite matrix (where D is a *diagonal* matrix) is inadvisable because even if the factors L and D are well-defined, they may contain entries that are larger than the original elements of A, thus amplifying rounding errors that arise during the computation. However, by using the block diagonal matrix B, which allows 2×2 blocks as well as 1×1 blocks on the diagonal, we can guarantee that the factorization (3.51) always exists and can be computed by a numerically stable process.

Example 3.2
The matrix

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \\ 3 & 2 & 3 & 4 \end{bmatrix}$$

can be written in the form (3.51) with $P = [e_1, e_4, e_3, e_2]$,

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{9} & \frac{2}{3} & 1 & 0 \\ \frac{2}{9} & \frac{1}{3} & 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 3 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & \frac{7}{9} & \frac{5}{9} \\ 0 & 0 & \frac{5}{9} & \frac{10}{9} \end{bmatrix}. \tag{3.52}$$

Note that both diagonal blocks in B are 2×2. Several algorithms for computing symmetric indefinite factorizations are discussed in Section **??**.

The symmetric indefinite factorization allows us to determine the *inertia* of a matrix, that is, the number of positive, zero, and negative eigenvalues. One can show that the inertia of B equals the inertia of A. Moreover, the 2×2 blocks in B are always constructed to have one positive and one negative eigenvalue. Thus the number of positive eigenvalues in A equals the number of positive 1×1 blocks plus the number of 2×2 blocks.

As for the Cholesky factorization, an indefinite symmetric factorization algorithm can be modified to ensure that the modified factors are the factors of a positive definite matrix. The strategy is first to compute the factorization (3.51), as well as the spectral decomposition $B = Q\Lambda Q^T$, which is very inexpensive to compute because B is block diagonal (see the exercises). Then we construct a modification matrix F such that

$$L(B + F)L^T$$

is sufficiently positive definite. Motivated by the modified spectral decomposition (3.43) we will choose a parameter $\delta > 0$ and define F to be

$$F = Q \operatorname{diag}(\tau_i) Q^T, \quad \tau_i = \begin{cases} 0, & \lambda_i \geq \delta, \\ \delta - \lambda_i, & \lambda_i < \delta, \end{cases} \qquad i = 1, 2, \ldots, n, \tag{3.53}$$

where λ_i are the eigenvalues of B. The matrix F is thus the modification of minimum Frobenius norm that ensures that all eigenvalues of the modified matrix $B + F$ are no less than δ. This strategy therefore modifies the factorization (3.51) as follows:

$$P(A + E)P^T = L(B + F)L^T, \qquad \text{where } E = P^T L F L^T P^T;$$

note that E will not be diagonal, in general. Hence, in contrast to the modified Cholesky approach, this modification strategy changes the entire matrix A and not just its diagonal. The aim of strategy (3.53) is that the modified matrix satisfies $\lambda_{\min}(A + E) \approx \delta$ whenever the original matrix A has $\lambda_{\min}(A) < \delta$. It is not clear, however, whether it always comes close to attaining this goal.

3.5 Step-Length Selection Algorithms

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

$$\phi(\alpha) = f(x_k + \alpha p_k), \tag{3.54}$$

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

$$\alpha_k = -\frac{\nabla f_k^T p_k}{p_k^T Q p_k}. \tag{3.55}$$

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 (3.54) as

$$\phi(\alpha_k) \leq \phi(0) + c_1 \alpha_k \phi'(0), \tag{3.56}$$

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

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

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

$$\phi_q(\alpha) = \left(\frac{\phi(\alpha_0) - \phi(0) - \alpha_0 \phi'(0)}{\alpha_0^2} \right) \alpha^2 + \phi'(0)\alpha + \phi(0). \tag{3.57}$$

(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

$$\alpha_1 = -\frac{\phi'(0)\alpha_0^2}{2\left[\phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0\right]}. \tag{3.58}$$

If the sufficient decrease condition (3.56) 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 (3.56) 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 8. 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 $< \alpha_i$ 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 [34, 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

$$\alpha_{i+1} = \alpha_i - (\alpha_i - \alpha_{i-1}) \left[\frac{\phi'(\alpha_i) + d_2 - d_1}{\phi'(\alpha_i) - \phi'(\alpha_{i-1}) + 2d_2} \right], \tag{3.59}$$

with

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

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 usually produces a quadratic rate of convergence of the iteration (3.59) 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 $\phi'(0) = \nabla f_{k-1}^T p_{k-1}$ and to define α_0 to be its minimizer. This strategy yields

$$\alpha_0 = \frac{2(f_k - f_{k-1})}{\phi'(0)}. \tag{3.60}$$

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 (3.60) 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 3.6 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 3.5 (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**(α_{i-1}, α_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**(α_i, α_{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 3.6 (zoom)
repeat
Interpolate (using quadratic, cubic, or bisection) to find
a trial step length α_j between α_{lo} and α_{hi};
Evaluate $\phi(\alpha_j)$;

```
    if φ(α_j) > φ(0) + c_1 α_j φ'(0) or φ(α_j) ≥ φ(α_lo)
        α_hi ← α_j;
    else
        Evaluate φ'(α_j);
        if |φ'(α_j)| ≤ −c_2 φ'(0)
            Set α_* ← α_j and stop;
        if φ'(α_j)(α_hi − α_lo) ≥ 0
            α_hi ← α_lo;
        α_lo ← α_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 $x_{\rm lo}$, then we set $\alpha_{\rm lo} \leftarrow \alpha_j$ to maintain condition (b). If this results in a violation of condition (c), we remedy the situation by setting $\alpha_{\rm hi}$ to the old value of $\alpha_{\rm lo}$. The reader should sketch some graphs to illustrate the workings of **zoom**! [2]

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 [33, 193]. 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 precision, 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 [77], Lemaréchal [165], Fletcher [86], and in particular Moré and Thuente [193].

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.

[2] Michael has suggested ways of making this line search description more precise. See his email. These changes are not easy, and certainly not high priority. Do if time permits

Notes and References

For an extensive discussion of line search termination conditions see Ortega and Rheinboldt [207]. 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 (3.29) 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 [10].

Some line search methods (see Goldfarb [115] and Moré and Sorensen [190]) 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 more thorough treatment of the modified Cholesky factorization see Gill, Murray, and Wright [107] or Dennis and Schnabel [150]. The modified Cholesky factorization based on Gershgorin disk estimates is described in Schnabel and Eskow [248]. The modified indefinite factorization is from Cheng and Higham [48].

Another strategy for implementing a line search Newton method when the Hessian contains negative eigenvalues is to compute a direction of negative curvature and use it to define the search direction (see Moré and Sorensen [190] and Goldfarb [115]).

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, [65, 33].

Our discussion of interpolation follows Dennis and Schnabel [77], and the algorithm for finding a step length satisfying the strong Wolfe conditions can be found in Fletcher [86].

3.6 Exercises

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 $x_0 = (-1.2, 1)$.

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

3. Show that the one-dimensional minimizer of a strongly convex quadratic function is given by (3.55).

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

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

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.

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

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

and

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

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

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

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).

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.

10. Compute the eigenvalues of the 2 diagonal blocks of (3.52), and verify that each block has a positive and a negative eigenvalue. Then compute the eigenvalues of A and verify that its inertia is the same as that of B.

11. Describe the effect that the modified Cholesky factorization of Algorithm **??** would have on the Hessian $\nabla^2 f(x_k) = \text{diag}(-2, 12, 4)$.

12. Consider a block diagonal matrix B with 1×1 and 2×2 blocks. Show that the eigenvalues and eigenvectors of B can be obtained by computing the spectral decomposition of each diagonal block separately.

13. Show that the quadratic function that interpolates $\phi(0)$, $\phi'(0)$, and $\phi(\alpha_0)$ is given by (3.57). 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

$$\alpha_1 < \frac{\alpha_0}{2(1 - c_1)}.$$

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.

14. If $\phi(\alpha_0)$ is large, (3.58) 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.)

15. 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 $\frac{2}{3}\alpha_1$.