Globally Convergent Modifications of Newton's Method

In the last chapter, Newton's method was shown to be locally q-quadratically convergent. This means that when the current solution approximation is good enough, it will be improved rapidly and with relative ease. Unfortunately, it is not unusual to expend significant computational effort in getting close enough. In addition, the strategies for getting close constitute the major part of the program and the programming effort, and they can be sensitive to small differences in implementation.

This chapter will be devoted to the two major ideas for proceeding when the Newton step is unsatisfactory. Section 6.1 will set the framework for the class of algorithms we want to consider, and Section 6.2 will reintroduce the concept of a descent direction that we saw briefly in the proof of Lemma 4.3.1. Section 6.3 discusses the first major global approach, modern versions of the traditional idea of backtracking along the Newton direction if a full Newton step is unsatisfactory. Section 6.4 discusses the second major approach, based on estimating the region in which the local model, underlying Newton's method, can be trusted to adequately represent the function, and taking a step to approximately minimize the model in this region. Both approaches are derived initially for the unconstrained minimization problem; their application to solving systems of nonlinear equations is the topic of Section 6.5.

---

* For our definition of "globally convergent," see the last paragraph of Section 1.1.

6.1 THE QUASI-NEWTON FRAMEWORK

The basic idea in forming a successful nonlinear algorithm is to combine a globally convergent strategy with a fast local strategy in a way that derives the benefits of both. The framework for doing this, a slight expansion of the hybrid algorithm discussed in Chapter 2 (Algorithm 2.5.1), is outlined in Algorithm 6.1.1 below. The most important point is to try Newton's method, or some modification of it, first at each iteration. If it seems to be taking a reasonable step—for example, if f decreases in a minimization application—use it. If not, fall back on a step dictated by a global method. Such a strategy will always end up using Newton's method close to the solution and thus retain its fast local convergence rate. If the global method is chosen and incorporated properly, the algorithm will also be globally convergent. We will call an algorithm that takes this approach *quasi-Newton.*

ALGORITHM 6.1.1 QUASI-NEWTON ALGORITHM FOR NONLINEAR EQUATIONS OR UNCONSTRAINED OPTIMIZATION [for optimization, replace $F(x_k)$ by $\nabla f(x_k)$ and J_k by H_k]

Given $F\colon \mathbb{R}^n \longrightarrow \mathbb{R}^n$ continuously differentiable, and $x_0 \in \mathbb{R}^n$. At each iteration k:

1. Compute $F(x_k)$, if it is not already done, and decide whether to stop or continue.
2. Compute J_k to be $J(x_k)$, or an approximation to it.
3. Apply a factorization technique to J_k and estimate its condition number. If J_k is ill-conditioned, perturb it in an appropriate manner.
4. Solve $J_k s_k^N = -F(x_k)$.
5. Decide whether to take a Newton step, $x_{k+1} = x_k + s_k^N$, or to choose x_{k+1} by a global strategy. This step often furnishes $F(x_k)$ to step 1.

At this point we have covered steps 2, 3, and 4; step 5 is the topic of this chapter. At the end of this chapter, then, the reader could construct a complete quasi-Newton algorithm, with the exception of the stopping criteria, which are covered in Chapter 7. The secant updates of Chapters 8 and 9 are alternate versions of step 2.

We remind the reader that a modular system of algorithms for nonlinear equations and unconstrained minimization is provided in Appendix A, supply-

ing a variety of completely detailed quasi-Newton algorithms for the two problems. Algorithm 6.1.1 is the basis of the driver for these algorithms, which in the appendix is Algorithm D6.1.1 for minimization and Algorithm D6.1.3 for nonlinear equations. The main difference between Algorithm 6.1.1 above and the two drivers in the appendix is that the actual drivers contain a separate initialization section preceeding the iteration section.

The purpose, organization, and use of the modular system of algorithms is discussed at the beginning of Appendix A. We recommend reading this discussion now if you have not previously done so, and then referring back to the appendix for details as the algorithms arise in the text.

6.2 DESCENT DIRECTIONS

We start our discussion of global methods by considering the unconstrained minimization problem

$$\min_{x \in \mathbb{R}^n} f\colon \mathbb{R}^n \longrightarrow \mathbb{R}, \tag{6.2.1}$$

because there is a natural global strategy for minimization problems; it is to make sure that each step decreases the value of f. There are corresponding strategies for systems of nonlinear equations, in particular, making sure each step decreases the value of some norm of $F\colon \mathbb{R}^n \longrightarrow \mathbb{R}^n$. However, by using a norm, one is really transforming the problem of solving the system of equations $F(x) = 0$ into the problem of minimizing a function, such as

$$f \triangleq \tfrac{1}{2}\| F \|_2^2 : \mathbb{R}^n \longrightarrow \mathbb{R}. \tag{6.2.2}$$

Therefore, for this section and the next two we will discuss global strategies for unconstrained minimization. In Section 6.5 we will apply these strategies to systems of nonlinear equations, using the transformation (6.2.2).

The basic idea of a global method for unconstrained minimization is geometrically obvious: take steps that lead "downhill" for the function f. More precisely, one chooses a direction p from the current point x_c in which f decreases initially, and a new point x_+ in this direction from x_c such that $f(x_+) < f(x_c)$. Such a direction is called a *descent direction.* Mathematically, p is a descent direction from x_c if the directional derivative of f at x_c in the direction p is negative—that is, from Section 4.1, if

$$\nabla f(x_c)^T p < 0. \tag{6.2.3}$$

If (6.2.3) holds, then it is guaranteed that for sufficiently small positive δ, $f(x_c + \delta p) < f(x_c)$. Descent directions form the basis of some global methods for minimization and are important to all of them, and therefore they are the topic of this section.

So far, the only direction we have considered for minimization is the Newton direction $s^N = -H_c^{-1} \nabla f(x_c)$, where H_c is either $\nabla^2 f(x_c)$ or an approximation to it. Therefore, it is natural to ask whether the Newton direction is a descent direction. By (6.2.3), it is if and only if

$$\nabla f(x_c)^T s^N = -\nabla f(x_c)^T H_c^{-1} \nabla f(x_c) < 0,$$

which is true if H_c^{-1} or, equivalently, H_c is positive definite. This is the second reason why, in Algorithm 5.4.2, we coerce H_c to be positive definite if it isn't already. It guarantees not only a quadratic model with a unique minimizer but also that the resultant Newton direction will be a descent direction for the actual problem. This in turn guarantees that for sufficiently small steps in the Newton direction, the function value will be decreased. In all our methods for unconstrained minimization, the model Hessian H_c will be formed or coerced to make it positive definite.

A second natural question is: as long as one is taking steps in descent directions, what is the direction p in which f decreases most rapidly from x? The notion of direction needs to be made more explicit before we can answer this question, since, for a given perturbation vector p, the directional derivative of f in the direction p is directly proportional to the length of p. A reasonable way to proceed is to choose a norm $\|\cdot\|$, and define the direction of any p as $p/\|p\|$. When we speak of a direction p, we will assume that this normalization has been carried out. We can now pose our question about a most rapid descent direction for a given norm as

$$\min_{p \in \mathbb{R}^n} \nabla f(x)^T p \qquad \text{subject to } \|p\| = 1,$$

which, in the l_2 norm, has solution $p = -\nabla f(x)/\|\nabla f(x)\|_2$. Thus the negative of the gradient direction is the steepest downhill direction from x in the l_2 norm, and it is referred to as the *steepest-descent direction.*

A classic minimization algorithm due to Cauchy is based solely on the steepest-descent direction. Its theoretical form is given below.

ALGORITHM 6.2.1 METHOD OF STEEPEST DESCENT

Given $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ continuously differentiable, $x_0 \in \mathbb{R}^n$. At each iteration k:

find the lowest point of f in the direction $-\nabla f(x_k)$ from x_k—i.e., find λ_k that solves

$$\min_{\lambda_k > 0} f(x_k - \lambda_k \nabla f(x_k))$$

$$x_{k+1} := x_k - \lambda_k \nabla f(x_k)$$

This is not a computational method, because each step contains an exact one-dimensional minimization problem, but it can be implemented by doing an inexact minimization. In either case, under mild conditions it can be shown to converge to a local minimizer or saddle point of $f(x)$. [See, e.g., Goldstein (1967).] However, the convergence is only linear, and sometimes very slowly linear. Specifically, if Algorithm 6.2.1 converges to a local minimizer x_* where $\nabla^2 f(x_*)$ is positive definite, and $\text{ev}_{\max}$ and $\text{ev}_{\min}$ are the largest and smallest eigenvalues of $\nabla^2 f(x_*)$, then one can show that $\{x_k\}$ satisfies

$$\limsup_{k\to\infty} \frac{\|x_{k+1} - x_*\|}{\|x_k - x_*\|} \leq c, \qquad c \triangleq \left(\frac{\text{ev}_{\max} - \text{ev}_{\min}}{\text{ev}_{\max} + \text{ev}_{\min}}\right) \tag{6.2.4}$$

in a particular weighted l_2 norm, and that the bound on c is tight for some starting x_0. If $\nabla^2 f(x_*)$ is nicely scaled with $\text{ev}_{\max} \cong \text{ev}_{\min}$, then c will be very small and convergence will be fast, but if $\nabla^2 f(x_*)$ is even slightly poorly scaled—e.g., $\text{ev}_{\max} = 10^2\ \text{ev}_{\min}$—then c will be almost 1 and convergence may be very slow (see Figure 6.2.1). An example, where $c = 0.8$ and $\|x_{k+1} - x_*\| = c\|x_k - x_*\|$ at each iteration, is given in Example 6.2.2. It is any easy exercise to generalize this example to make c arbitrarily close to 1.

EXAMPLE 6.2.2 Let $f(x_1, x_2) = \frac{1}{2}x_1^2 + \frac{9}{2}x_2^2$. This is a positive definite quadratic with minimizer at $x_* = (0, 0)^T$ and

$$\nabla^2 f(x) = \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix} \qquad \text{for all } x.$$

Thus, the c given by (6.2.4) is $(9 - 1)/(9 + 1) = 0.8$. Also, the reader can verify that if $f(x)$ is a positive definite quadratic, the steepest descent step of Algorithm 6.2.1 is given by

$$x_{k+1} := x_k - \left(\frac{g^T g}{g^T \nabla^2 f(x_k) g}\right) g, \tag{6.2.5}$$

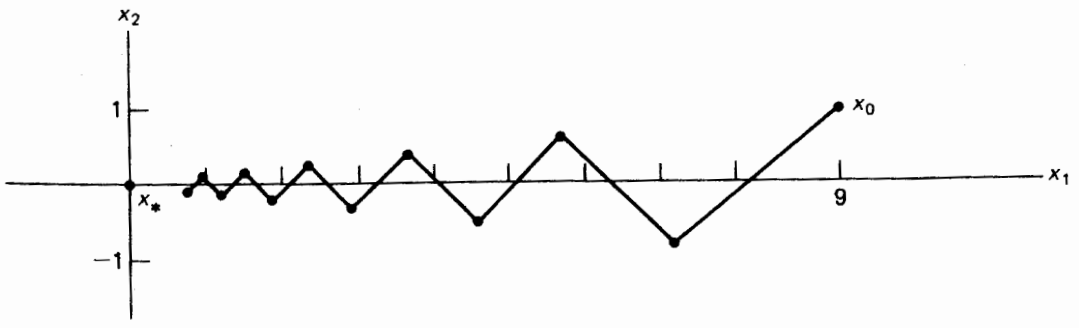


Figure 6.2.1 Method of steepest descent on a slightly poorly scaled quadratic

where $g = \nabla f(x_k)$. Using (6.2.5), it is easy to verify that the sequence of points generated by Algorithm 6.2.1 with the given $f(x)$ and $x_0 = (9, 1)^T$ is

$$x_k = \begin{pmatrix} 9 \\ (-1)^k \end{pmatrix} (0.8)^k, \qquad k = 1, 2, \ldots. \tag{6.2.6}$$

Therefore, $\| x_{k+1} - x_* \| / \| x_k - x_* \| = \| x_{k+1} \| / \| x_k \| = c$ in any l_p norm.

The sequence $\{x_k\}$ given in (6.2.6) is drawn in Figure 6.2.1.

Thus, the method of steepest descent should not be used as a computational algorithm in most circumstances, since a quasi-Newton algorithm will be far more efficient. However, when our global strategy must take steps much smaller than the Newton step, we will see in Section 6.4 that it may take steps in, or close to, the steepest descent direction.

We note finally that the steepest descent direction is very sensitive to changes in the scale of x, while the Newton direction is independent of such changes. For this reason, when we seem to use the steepest descent direction in our algorithms of Section 6.4, the corresponding implementations in the appendix will actually first premultiply it by a diagonal scaling matrix. We defer to Section 7.1 consideration of this important practical topic called scaling.

6.3 LINE SEARCHES

The first strategy we consider for proceeding from a solution estimate outside the convergence region of Newton's method is the method of line searches. This is really a very natural idea after the discussion of descent directions in Section 6.2.

The idea of a line-search algorithm is simple: given a descent direction p_k, we take a step in that direction that yields an "acceptable" x_{k+1}. That is,

at iteration k:

calculate a descent direction p_k,

set $x_{k+1} := x_k + \lambda_k p_k$ for some $\lambda_k > 0$ that makes x_{k+1} an acceptable next iterate. (6.3.1)

Graphically, this means selecting x_{k+1} by considering the half of a one-dimensional cross section of $f(x)$ in which $f(x)$ decreases initially from x_k, as depicted in Figure 6.3.1. However, rather than setting p_k to the steepest descent direction $-\nabla f(x_k)$ as in Algorithm 6.2.1, we will use the quasi-Newton direction $-H_k^{-1} \nabla f(x_k)$ discussed in Section 5.5, where $H_k = \nabla^2 f(x_k) + \mu_k I$ is positive definite with $\mu_k = 0$ if $\nabla^2 f(x_k)$ is safely positive definite. This will allow us to retain fast local convergence.

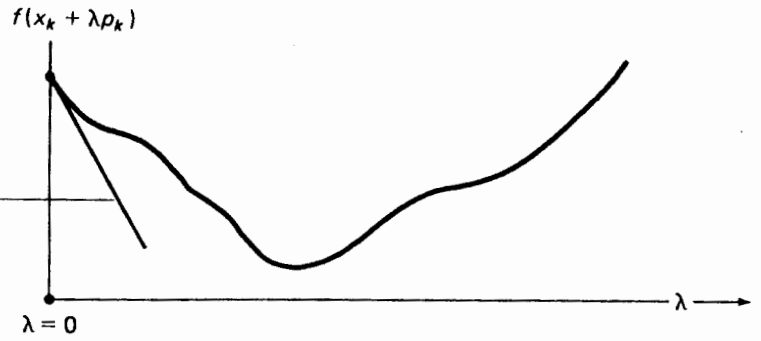


Figure 6.3.1 A cross section of $f(x)$: $\mathbb{R}^n \to \mathbb{R}$ from x_k in the direction p_k

The term "line search" refers to a procedure for choosing λ_k in (6.3.1). Until the mid 1960s the prevailing belief was that λ_k should be chosen to solve the one-dimensional minimization problem accurately. More careful computational testing has led to a complete turnabout, and in this section we will give weak acceptance criteria for $\{\lambda_k\}$ that lead to methods that perform as well in theory and better in practice.

The common procedure now is to try the full quasi-Newton step first and, if $\lambda_k = 1$ fails to satisfy the criterion in use, to backtrack in a systematic way along the direction defined by that step. *Computational experience has shown the importance of taking a full quasi-Newton step whenever possible.* Failure to do so leads to forfeiture of the advantage of Newton's method near the solution. Therefore, it is important that our procedure for choosing λ_k allow $\lambda_k = 1$ near the solution.

In the remainder of this section we discuss the choice of λ_k in theory and in practice. Since there is no compelling reason to confine our discussion to line searches in the quasi-Newton direction, we consider the search for $x_+ = x_c + \lambda_c p$ along a general descent direction p from the current solution estimate x_c.

While no step-acceptance rule will always be optimal, it does seem to be common sense to require that

$$f(x_{k+1}) < f(x_k). \tag{6.3.2}$$

It comes as no great surprise that this simple condition does not gurantee that $\{x_k\}$ will converge to a minimizer of f. We look now at two simple one-dimensional examples that show two ways that a sequence of iterates can satisfy (6.3.2) but fail to converge to a minimizer. These examples will guide us to useful step-acceptance criteria.

First, let $f(x) = x^2, x_0 = 2$. If we choose $\{p_k\} = \{(-1)^{k+1}\}$, $\{\lambda_k\} = \{2 + 3(2^{-(k+1)})\}$, then $\{x_k\} = \{2, -\frac{3}{2}, \frac{5}{4}, -\frac{9}{8}, \dots\} = \{(-1)^k(1 + 2^{-k})\}$, each p_k is a descent direction from x_k, and $f(x_k)$ is monotonically decreasing with

$$\lim_{k \to \infty} f(x_k) = 1$$

[see Figure 6.3.2(a)]. Of course this is not a minimum of any sort for f, and furthermore $\{x_k\}$ has limit points ± 1, so it does not converge.

Now consider the same function with the same initial estimate, and let us take $\{p_k\} = \{-1\}$, $\{\lambda_k\} = \{2^{-k+1}\}$. Then $\{x_k\} = \{2, \frac{3}{2}, \frac{5}{4}, \frac{9}{8}, \dots\} = \{1 + 2^{-k}\}$, each p_k is again a descent direction, $f(x_k)$ decreases monotonically, and $\lim_{k\to\infty} x_k = 1$, which again is not a minimizer of f [see Figure 6.3.2(b)].

The problem in the first case is that we achieved very small decreases in f values relative to the lengths of the steps. We can fix this by requiring that the average rate of decrease from $f(x_c)$ to $f(x_+)$ be at least some prescribed fraction of the initial rate of decrease in that direction; that is, we pick an $\alpha \in (0, 1)$ and choose λ_c from among those $\lambda > 0$ that satisfy

$$f(x_c + \lambda p) \le f(x_c) + \alpha\lambda \nabla f(x_c)^T p \tag{6.3.3a}$$

(see Figure 6.3.3). Equivalently, λ_c must be chosen so that

$$f(x_+) \le f(x_c) + \alpha \nabla f(x_c)^T (x_+ - x_c). \tag{6.3.3b}$$

It can be verified that this precludes the first unsuccessful choice of points but not the second.

The problem in the second example is that the steps are too small, relative to the initial rate of decrease of f. There are various conditions that ensure sufficiently large steps; we will present one that will also be useful to us in Chapter 9. We will require that the rate of decrease of f in the direction p at x_+ be larger than some prescribed fraction of the rate of decrease in the direction p at x_c—that is,

$$\nabla f(x_+)^T p \triangleq \nabla f(x_c + \lambda_c p)^T p \ge \beta \nabla f(x_c)^T p, \tag{6.3.4a}$$

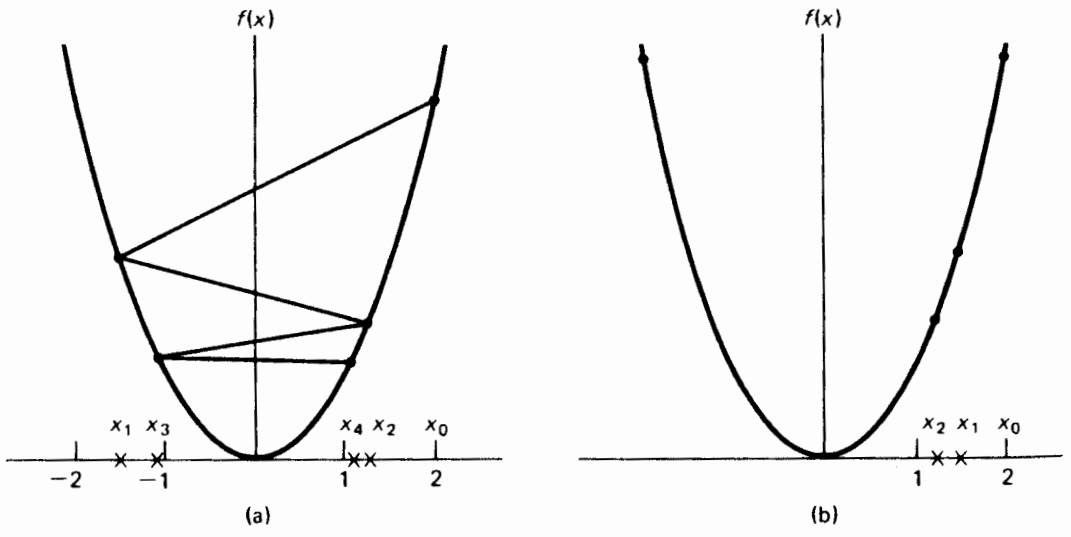


Figure 6.3.2 Monotonically decreasing sequences of iterates that don't converge to the minimizer

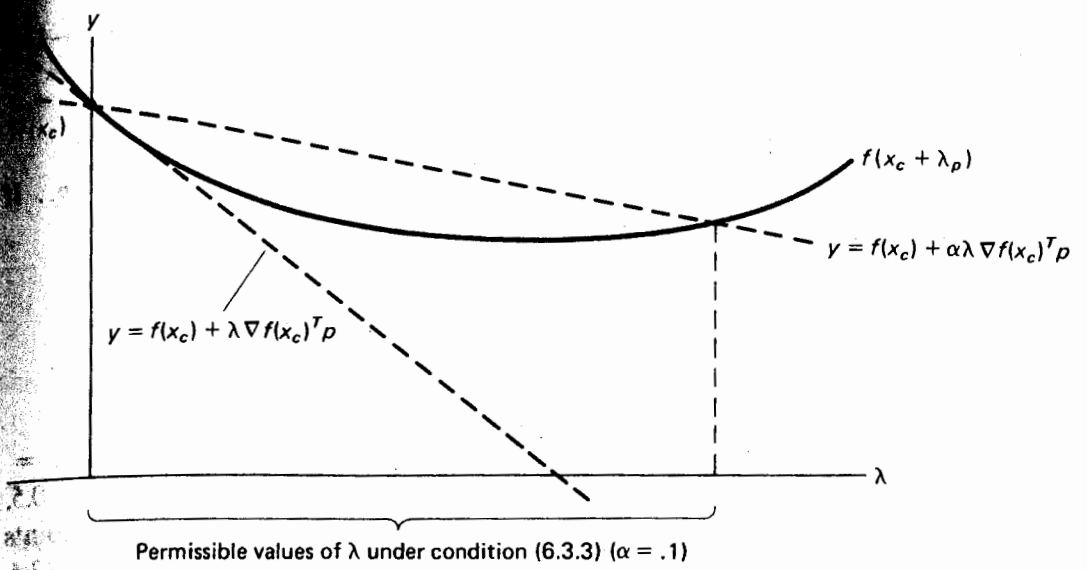


Figure 6.3.3 Permissible values of λ under condition (6.3.3) ($\alpha = 0.1$)

or equivalently,

$$\nabla f(x_+)^T(x_+ - x_c) \geq \beta \nabla f(x_c)^T(x_+ - x_c) \tag{6.3.4b}$$

for some fixed constant $\beta \in (\alpha, 1)$ (see Figure 6.3.4). The condition $\beta > \alpha$ guarantees that (6.3.3) and (6.3.4) can be satisfied simultaneously. In practice, (6.3.4) generally is not needed because the use of a backtracking strategy avoids excessively small steps. An alternative condition to (6.3.4) is given in Exercise 7.

Conditions (6.3.3) and (6.3.4) are based on work of Armijo (1966) and Goldstein (1967). Example 6.3.1 demonstrates the effect of these conditions on a simple function. In Subsection 6.3.1 we prove powerful convergence results for any algorithm obeying these conditions. Subsection 6.3.2 discusses a practical line-search algorithm.

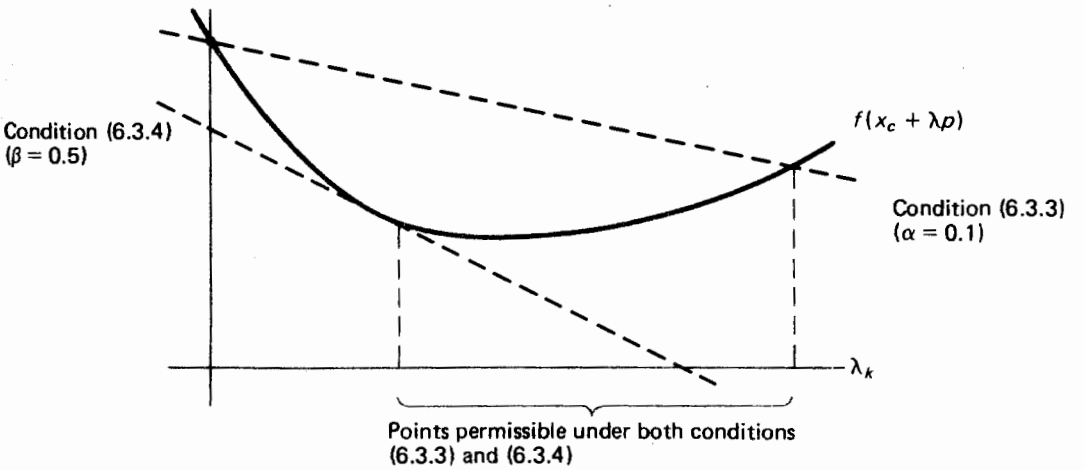


Figure 6.3.4 The two line search conditions

EXAMPLE 6.3.1 Let $f(x_1, x_2) = x_1^4 + x_1^2 + x_2^2$, $x_c = (1, 1)^T$, $p_c = (-3, -1)^T$, and let $\alpha = 0.1$ in (6.3.3), $\beta = 0.5$ in (6.3.4). Since

$$\nabla f(x_c)^T p_c = (6, 2)(-3, -1)^T = -20,$$

p_c is a descent direction for $f(x)$ from x_c. Now consider $x(\lambda) = x_c + \lambda p_c$. If $x_+ = x(1) = (-2, 0)^T$,

$$\nabla f(x_+)^T p_c = (-36, 0)(-3, -1)^T = 108 > -10 = \beta \nabla f(x_c)^T p_c,$$

so that x_+ satisfies (6.3.4), but

$$f(x_+) = 20 > 1 = f(x_c) + \alpha \nabla f(x_c)^T p_c,$$

so that x_+ doesn't satisfy (6.3.3). Similarly, the reader can verify that $x_+ = x(0.1) = (0.7, 0.9)^T$ satisfies (6.3.3) but not (6.3.4), and that $x_+ = x(0.5) = (-0.5, 0.5)^T$ satisfies both (6.3.3) and (6.3.4). These three points correspond to points to the right of, to the left of, and in the "permissible" region in Figure 6.3.4, respectively.

6.3.1 CONVERGENCE RESULTS FOR PROPERLY CHOSEN STEPS

Using conditions (6.3.3) and (6.3.4), we can prove some amazingly powerful results: that given any direction p_k such that $\nabla f(x_k)^T p_k < 0$, there exist $\lambda_k > 0$ satisfying (6.3.3) and (6.3.4); that any method that generates a sequence $\{x_k\}$ obeying $\nabla f(x_k)^T (x_{k+1} - x_k) < 0$, (6.3.3), and (6.3.4) at each iteration is essentially globally convergent; and that close to a minimizer of f where $\nabla^2 f$ is positive definite, Newton steps satisfy (6.3.3) and (6.3.4). This means we can readily create algorithms that are globally convergent and, by always trying the Newton step first, are also quickly locally convergent on most functions.

Theorem 6.3.2, due to Wolfe (1969, 1971), shows that given any descent direction p_k, there exist points $x_k + \lambda_k p_k$ satisfying (6.3.3) and (6.3.4).

THEOREM 6.3.2 Let $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ be continuously differentiable on $\mathbb{R}^n$. Let $x_k \in \mathbb{R}^n$, $p_k \in \mathbb{R}^n$ obey $\nabla f(x_k)^T p_k < 0$, and assume $\{f(x_k + \lambda p_k) \mid \lambda > 0\}$ is bounded below. Then if $0 < \alpha < \beta < 1$, there exist $\lambda_u > \lambda_l > 0$ such that $x_{k+1} = x_k + \lambda_k p_k$ satisfies (6.3.3) and (6.3.4) if $\lambda_k \in (\lambda_l, \lambda_u)$.

Proof. Since $\alpha < 1$, $f(x_k + \lambda p_k) < f(x_k) + \alpha\lambda \nabla f(x_k)^T p_k$ for all $\lambda > 0$ sufficiently small. Since $f(x_k + \lambda p_k)$ is also bounded below, there exists some smallest positive $\hat{\lambda}$ such that

$$f(\hat{x}) = f(x_k + \hat{\lambda} p_k) = f(x_k) + \alpha\hat{\lambda} \nabla f(x_k)^T p_k. \tag{6.3.5}$$

Thus any $x \in (x_k, \hat{x})$ satisfies (6.3.3). By the mean value theorem, there

exists $\bar{\lambda} \in (0, \hat{\lambda})$ such that

$$f(\hat{x}) - f(x_k) = \nabla f(x_k + \bar{\lambda} p_k)^T (\hat{x} - x_k)$$
$$= \nabla f(x_k + \bar{\lambda} p_k)^T \hat{\lambda} p_k, \tag{6.3.6}$$

and so from (6.3.5) and (6.3.6),

$$\nabla f(x_k + \bar{\lambda} p_k)^T p_k = \alpha \nabla f(x_k)^T p_k > \beta \nabla f(x_k)^T p_k \tag{6.3.7}$$

since $\alpha < \beta$ and $\nabla f(x_k)^T p_k < 0$. By the continuity of ∇f, (6.3.7) still holds for λ in some interval (λ_l, λ_u) about $\bar{\lambda}$. Therefore, if we restrict (λ_l, λ_u) to be in $(0, \hat{\lambda})$, $x_{k+1} = x_k + \lambda_k p_k$ satisfies (6.3.3) and (6.3.4) for any $\lambda_k \in (\lambda_l, \lambda_u)$. □

Theorem 6.3.3 also is due to Wolfe (1969, 1971). It shows that if any sequence $\{x_k\}$ is generated to obey $\nabla f(x_k)^T (x_{k+1} - x_k) < 0$, (6.3.3), and (6.3.4), then, unless the angle between $\nabla f(x_k)$ and $(x_{k+1} - x_k)$ converges to 90° as $k \longrightarrow \infty$, either the gradient converges to 0, or f is unbounded below. Of course, both may be true. We comment below that the case where the angle between $\nabla f(x_k)$ and $(x_{k+1} - x_k)$ approaches 90° can be prevented by the algorithm.

THEOREM 6.3.3 Let $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ be continuously differentiable on $\mathbb{R}^n$, and assume there exists $\gamma \geq 0$ such that

$$\| \nabla f(z) - \nabla f(x) \|_2 \leq \gamma \| z - x \|_2 \tag{6.3.8}$$

for every $x, z \in \mathbb{R}^n$. Then, given any $x_0 \in \mathbb{R}^n$, either f is unbounded below, or there exists a sequence $\{x_k\}$, $k = 0, 1, \ldots$, obeying (6.3.3), (6.3.4), and either

$$\nabla f(x_k)^T s_k < 0 \tag{6.3.9}$$

or

$$\nabla f(x_k) = 0 \quad \text{and} \quad s_k = 0,$$

for each $k \geq 0$, where

$$s_k \triangleq x_{k+1} - x_k.$$

Furthermore, for any such sequence, either

(a) $\nabla f(x_k) = 0$ for some $k \geq 0$, or

(b) $\lim_{k \to \infty} f(x_k) = -\infty$, or

(c) $\lim_{k \to \infty} \dfrac{\nabla f(x_k)^T s_k}{\| s_k \|_2} = 0.$

Proof. For each k, if $\nabla f(x_k) = 0$, then (a) holds and the sequence is constant subsequently. If $\nabla f(x_k) \neq 0$, then there exists p_k—e.g., $p_k = -\nabla f(x_k)$—such that $f(x_k)^T p_k < 0$. By Theorem 6.3.2, either f is unbounded below, or there exists $\lambda_k > 0$ such that $x_{k+1} = x_k + \lambda_k p_k$ satisfies (6.3.3) and (6.3.4). In order to simplify notation, let us assume that $\| p_k \|_2 = 1$, so $\lambda_k = \| s_k \|_2$. This constitutes no loss of generality.

So far we see that either f is unbounded below, or $\{x_k\}$ exists, and either $\{x_k\}$ satisfies (a) or else $s_k \neq 0$ for every k. We must now show that if no term of $\{s_k\}$ is zero, then (b) or (c) must hold. It will be useful to have

$$\sigma_k \triangleq \frac{\nabla f(x_k)^T s_k}{\| s_k \|_2}.$$

By (6.3.3) and $\lambda_k \sigma_k < 0$ for every k, for any $j > 0$,

$$\begin{aligned} f(x_j) - f(x_0) &= \sum_{k=0}^{j-1} f(x_{k+1}) - f(x_k) \\ &\leq \sum_{k=0}^{j-1} \alpha \nabla f(x_k)^T s_k \\ &= \alpha \sum_{k=0}^{j-1} \lambda_k \sigma_k < 0. \end{aligned}$$

Hence, either

$$\lim_{j \to \infty} f(x_j) = -\infty \quad \text{or} \quad -\sum_{k=0}^{\infty} \lambda_k \sigma_k < \infty$$

is convergent.

In the first case (b) is true and we are finished, so we consider the second. In particular, we deduce that $\lim_{k \to \infty} \lambda_k \sigma_k = 0$. Now we want to conclude that $\lim_{k \to \infty} \sigma_k = 0$, and so we need to use condition (6.3.4), since it was imposed to ensure that the steps don't get too small.

We have for each k that

$$\nabla f(x_{k+1})^T s_k \geq \beta \nabla f(x_k)^T s_k$$

and so

$$[\nabla f(x_{k+1}) - \nabla f(x_k)]^T s_k \geq (\beta - 1) \nabla f(x_k)^T s_k > 0,$$

by (6.3.9) and $\beta < 1$. Applying the Cauchy-Schwarz inequality and (6.3.8) to the left side of the last equation, and using the definition of σ_k, gives us

$$0 < (\beta - 1)\lambda_k \sigma_k \leq \gamma \| s_k \|^2 = \gamma \lambda_k^2,$$

so

$$\lambda_k \geq \frac{\beta - 1}{\gamma} \sigma_k > 0$$

and

$$\lambda_k \sigma_k \leq \frac{\beta - 1}{\gamma} \sigma_k^2 < 0.$$

Thus

$$0 = \lim_{k \to \infty} \lambda_k \sigma_k \leq \frac{\beta - 1}{\gamma} \lim_{k \to \infty} \sigma_k^2 \leq 0,$$

which shows that

$$\lim_{k \to \infty} \sigma_k = 0$$

[i.e., (c) is true] and completes the proof. □

Note that while Theorem 6.3.3 applies readily to any line-search algorithm, it is completely independent of the method for selecting the descent directions or the step lengths. Therefore, this theorem gives sufficient conditions for global convergence, in a weak sense, of any optimization algorithm, including the model-trust region algorithms of Section 6.4. Furthermore, while the Lipschitz condition (6.3.8) is assumed on all of $\mathbb{R}^n$, it is used only in a neighborhood of the solution x_*. Finally, although $\{\sigma_k\} \longrightarrow 0$ in Theorem 6.3.3 does not necessarily imply $\{\nabla f(x_k)\} \longrightarrow 0$, it does as long as the angle between $\nabla f(x_k)$ and s_k is bounded away from 90°. This can easily be achieved in practice. For example, in a quasi-Newton line-search algorithm where $p_k = -H_k^{-1} \nabla f(x_k)$ and H_k is positive definite, all that is needed is that the condition numbers of $\{H_k\}$ are uniformly bounded above. Thus, Theorem 6.3.3 can be viewed as implying global covergence toward $f = -\infty$ or $\nabla f = 0$, although the conditions are too weak to imply that $\{x_k\}$ converges.

Theorem 6.3.4, due to Dennis and Moré (1974), shows that our global strategy will permit full quasi-Newton steps $x_{k+1} := x_k - H_k^{-1} \nabla f(x_k)$ close to a minimizer of f as long as $-H_k^{-1} \nabla f(x_k)$ is close enough to the Newton step.

THEOREM 6.3.4 Let $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ be twice continuously differentiable in an open convex set D, and assume that $\nabla^2 f \in \text{Lip}_\gamma(D)$. Consider a sequence $\{x_k\}$ generated by $x_{k+1} := x_k + \lambda_k p_k$, where $\nabla f(x_k)^T p_k < 0$ for all k and λ_k is chosen to satisfy (6.3.3) with an $\alpha < \frac{1}{2}$, and (6.3.4). If $\{x_k\}$ converges to a point $x_* \in D$ at which $\nabla^2 f(x_*)$ is positive definite, and if

$$\lim_{k \to \infty} \frac{\| \nabla f(x_k) + \nabla^2 f(x_k) p_k \|_2}{\| p_k \|_2} = 0, \tag{6.3.10}$$

then there is an index $k_0 \geq 0$ such that for all $k \geq k_0$, $\lambda_k = 1$ is admissible. Furthermore, $\nabla f(x_*) = 0$, and if $\lambda_k = 1$ for all $k \geq k_0$, then $\{x_k\}$ converges q-superlinearly to x_*.

Proof. The proof is really just a generalization of the easy exercise that if $f(x)$ is a positive definite quadratic and $p_k = -\nabla^2 f(x_k)^{-1}\,\nabla f(x_k)$, then $\nabla f(x_k)^T p_k < 0$ and $x_{k+1} = x_k + p_k$ satisfies (6.3.3) for any $\alpha \le \frac{1}{2}$, and (6.3.4) for any $\beta \ge 0$. Some readers may wish to skim or skip it, as the details are not too illuminating. We set

$$\rho_k = \frac{\| \nabla f(x_k) + \nabla^2 f(x_k) p_k \|}{\| p_k \|}, \tag{6.3.11}$$

where throughout this proof, $\| \cdot \|$ means $\| \cdot \|_2$. First we show that

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

By a now familiar argument, if $\bar{x}$ is near enough x_*, then $\nabla^2 f(\bar{x})^{-1}$ exists and is positive definite, and if $\mu^{-1} = \| \nabla^2 f(x_*)^{-1} \|$, then $\frac{1}{2}\mu^{-1} \le \| \nabla^2 f(\bar{x})^{-1} \| \le 2\mu^{-1}$. Therefore, since

$$-\frac{\nabla f(x_k)^T p_k}{\| p_k \|} = \frac{p_k^T \nabla^2 f(x_k) p_k}{\| p_k \|} - \frac{(\nabla f(x_k) + \nabla^2 f(x_k) p_k)^T p_k}{\| p_k \|}$$

and $p_k^T \nabla^2 f(x_k) p_k > \frac{1}{2}\mu \| p_k \|^2$, (6.3.11) and (6.3.10) show that for k sufficiently large,

$$-\frac{\nabla f(x_k)^T p_k}{\| p_k \|} \ge (\tfrac{1}{2}\mu - \rho_k) \| p_k \| \ge \tfrac{1}{4}\mu \| p_k \|. \tag{6.3.12}$$

Since from Theorem 6.3.3 we have

$$\lim_{k \to \infty} \frac{\nabla f(x_k)^T p_k}{\| p_k \|} = 0,$$

(6.3.12) implies

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

We now show that $\lambda_k = 1$ satisfies (6.3.3), for all k greater than or equal to some k_0. From the mean value theorem, for some $\bar{x}_k \in [x_k, x_k + p_k]$,

$$f(x_k + p_k) - f(x_k) = \nabla f(x_k)^T p_k + \tfrac{1}{2} p_k^T \nabla^2 f(\bar{x}_k) p_k,$$

or

$$\begin{aligned} f(x_k + p_k) - f(x_k) - \tfrac{1}{2}\nabla f(x_k)^T p_k &= \tfrac{1}{2}(\nabla f(x_k) + \nabla^2 f(\bar{x}_k) p_k)^T p_k \\ &= \tfrac{1}{2}(\nabla f(x_k) + \nabla^2 f(x_k) p_k)^T p_k + \tfrac{1}{2} p_k^T [\nabla^2 f(\bar{x}_k) - \nabla^2 f(x_k)] p_k \\ &\le \tfrac{1}{2}(\rho_k + \gamma \| p_k \|) \| p_k \|^2 \end{aligned} \tag{6.3.13}$$

because of (6.3.11) and the Lipschitz continuity of $\nabla^2 f$. Now choose k_0 so

that for $k \geq k_0$, (6.3.12) holds and

$$\rho_k + \gamma \| p_k \| \leq \tfrac{1}{4}\mu \cdot \min(\beta, 1 - 2\alpha). \tag{6.3.14}$$

If $k \geq k_0$, $\lambda_k = 1$ satisfies (6.3.3) because from (6.3.13), (6.3.12), and (6.3.14)

$$\begin{aligned} f(x_k + p_k) - f(x_k) &\leq \tfrac{1}{2} \nabla f(x_k)^T p_k + \tfrac{1}{8}\mu \cdot (1 - 2\alpha) \| p_k \|^2 \\ &\leq \tfrac{1}{2}(1 - (1 - 2\alpha)) \nabla f(x_k)^T p_k \\ &= \alpha \nabla f(x_k)^T p_k . \end{aligned}$$

To show that (6.3.4) is satisfied by $\lambda_k = 1$ for $k \geq k_0$, we use the mean value theorem again to get, for some $z_k \in (x_k, x_k + p_k)$, that

$$\begin{aligned} \nabla f(x_k + p_k)^T p_k &= (\nabla f(x_k) + \nabla^2 f(z_k) p_k)^T p_k \\ &= (\nabla f(x_k) + \nabla^2 f(x_k) p_k)^T p_k \\ &\quad + p_k^T (\nabla^2 f(z_k) - \nabla^2 f(x_k)) p_k \end{aligned}$$

so

$$\begin{aligned} |\nabla f(x_k + p_k)^T p_k| &\leq \rho_k \| p_k \|^2 + \gamma \| p_k \|^3 \\ &\leq \frac{\mu\beta}{4} \| p_k \|^2 \\ &\leq -\beta \nabla f(x_k)^T p_k \end{aligned}$$

by (6.3.11), the Lipschitz continuity of $\nabla^2 f$, (6.3.14), and (6.3.12). This yields $\nabla f(x_k + p_k)^T p_k \geq \beta \nabla f(x_k)^T p_k$ for $k \geq k_0$. Thus $\lambda_k = 1$ eventually satisfies (6.3.3, 6.3.4), and so it is admissible. It is an easy consequence of (6.3.10) and $\lim_{k \to \infty} p_k = 0$ that $\nabla f(x_*) = 0$. This leaves only the proof of q-superlinear convergence if $\lambda_k = 1$ is chosen for all but finitely many terms. We postpone this until it follows from a much more general result in Chapter 8. □

Taken together, the conclusions of Theorems 6.3.3 and 6.3.4 are quite remarkable. They say that if f is bounded below, then the sequence $\{x_k\}$ generated by any algorithm that takes descent steps whose angles with the gradients are bounded away from 90°, and that satisfy conditions (6.3.3) and (6.3.4), will obey

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

Furthermore, if any such algorithm tries a Newton or quasi-Newton step first at each iteration, then $\{x_k\}$ will also converge q-quadratically or q-superlinearly to a local minimizer x_* if any x_k is sufficiently close to x_*, and if local convergence assumptions for the Newton or quasi-Newton method are met. These results are all we will need for the global convergence of the algorithms we discuss in the remainder of this book. (See also Exercise 25.)

6.3.2 STEP SELECTION BY BACKTRACKING

We now specify how our line-search algorithm will choose λ_k. As we have stated, the modern strategy is to start with $\lambda_k = 1$, and then, if $x_k + p_k$ is not acceptable, "backtrack" (reduce λ_k) until an acceptable $x_k + \lambda_k p_k$ is found. The framework of such an algorithm is given below. Recall that condition (6.3.4) is not implemented because the backtracking strategy avoids excessively small steps. (See also Exercise 25.)

ALGORITHM 6.3.5 BACKTRACKING LINE-SEARCH FRAMEWORK

Given $\alpha \in (0, \frac{1}{2})$, $0 < l < u < 1$

$\lambda_k = 1$;

while $f(x_k + \lambda_k p_k) > f(x_k) + \alpha\lambda_k \nabla f(x_k)^T p_k$, do

$\lambda_k := \rho\lambda_k$ for some $\rho \in [l, u]$;

(*ρ is chosen anew each time by the line search*)

$x_{k+1} := x_k + \lambda_k p_k$;

In practice, α is set quite small, so that hardly more than a decrease in function value is required. Our algorithm uses $\alpha = 10^{-4}$. Now just the strategy for reducing λ_k (choosing ρ) remains. Let us define

$$\hat{f}(\lambda) \triangleq f(x_k + \lambda p_k),$$

the one-dimensional restriction of f to the line through x_k in the direction p_k. If we need to backtrack, we will use our most current information about $\hat{f}$ to model it, and then take the value of λ that minimizes this model as our next value of λ_k in Algorithm 6.3.5.

Initially, we have two pieces of information about $\hat{f}(\lambda)$,

$$\hat{f}(0) = f(x_k) \quad \text{and} \quad \hat{f}'(0) = \nabla f(x_k)^T p_k. \tag{6.3.15}$$

After calculating $f(x_k + p_k)$, we also know that

$$\hat{f}(1) = f(x_k + p_k), \tag{6.3.16}$$

so if $f(x_k + p_k)$ does not satisfy (6.3.3) [i.e., $\hat{f}(1) > \hat{f}(0) + \alpha f'(0)$], we model $\hat{f}(\lambda)$ by the one-dimensional quadratic satisfying (6.3.15) and (6.3.16),

$$\hat{m}_q(\lambda) = [\hat{f}(1) - \hat{f}(0) - \hat{f}'(0)]\lambda^2 + \hat{f}'(0)\lambda + \hat{f}(0),$$

calculate the point

$$\hat{\lambda} = \frac{-\hat{f}'(0)}{2[\hat{f}(1) - \hat{f}(0) - \hat{f}'(0)]} \tag{6.3.17}$$

which $\hat{m}_q'(\hat{\lambda}) = 0$. Now

$$m_q''(\lambda) = 2[\hat{f}(1) - \hat{f}(0) - \hat{f}'(0)] > 0,$$

since $\hat{f}(1) > \hat{f}(0) + \alpha\hat{f}'(0) > \hat{f}(0) + \hat{f}'(0)$. Thus $\hat{\lambda}$ minimizes $\hat{m}_q(\lambda)$. Also $\hat{\lambda} > 0$, because $f'(0) < 0$. Therefore we take $\hat{\lambda}$ as our new value of λ_k in Algorithm 6.3.5 (see Figure 6.3.5). Note that since $\hat{f}(1) > \hat{f}(0) + \alpha\hat{f}'(0)$, we have

$$\hat{\lambda} < \frac{1}{2(1-\alpha)}.$$

In fact, if $\hat{f}(1) \geq \hat{f}(0)$, then $\hat{\lambda} \leq \frac{1}{2}$. Thus, (6.3.17) gives a useful implicit upper bound of $u \approx \frac{1}{2}$ on the first value of ρ in Algorithm 6.3.5. On the other hand, if $\hat{f}(1)$ is much larger than $\hat{f}(0)$, $\hat{\lambda}$ can be very small. We probably do not want to decrease λ_k too much based on this information, since probably it indicates that $\hat{f}(\lambda)$ is poorly modeled by a quadratic in this region, so we impose a lower bound of $l = \frac{1}{10}$ in Algorithm 6.3.5. This means that at the first backtrack at each iteration, if $\hat{\lambda} \leq 0.1$, then we next try $\lambda_k = \frac{1}{10}$.

EXAMPLE 6.3.6 Let $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$, and x_c and p be given by Example 6.3.1. Since $f(x_c) = 3$ and $f(x_c + p) = 20$, $x_c + p$ is not acceptable and a backtrack is needed. Then $\hat{m}_q(0) = f(x_c) = 3$, $\hat{m}_q'(0) = \nabla f(x_c)^T p = -20$, $\hat{m}_q(1) = f(x_c + p) = 20$, and (6.3.17) gives $\hat{\lambda} = \frac{10}{37} \cong 0.270$. Now $x_c + \hat{\lambda}p \cong (0.189, 0.730)^T$ satisfies condition (6.3.3) with $\alpha = 10^{-4}$, since $f(x_c + \hat{\lambda}p) \cong 0.570 \leq 2.99946 = f(x_c) + \alpha\lambda\ \nabla f(x_c)^T p$. Therefore $x_+ := x_c + \hat{\lambda}p$. Incidentally, the minimum of $f(x_c + \lambda p)$ occurs at $\lambda_* \cong 0.40$.

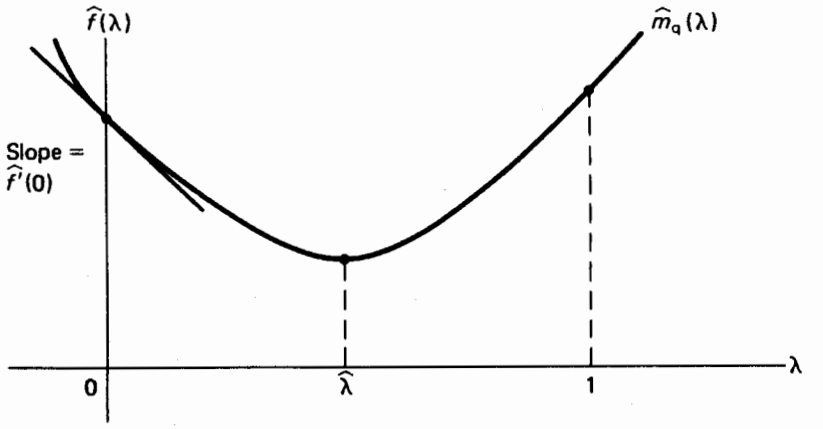


Figure 6.3.5 Backtracking at the first iteration, using a quadratic model

Now suppose $\hat{f}(\lambda_k) = f(x_k + \lambda_k p_k)$ doesn't satisfy (6.3.3). In this case we need to backtrack again. Although we could use a quadratic model as we did on the first backtrack, we now have four pieces of information about $\hat{f}(\lambda)$. So at this and any subsequent backtrack during the current iteration, we use a cubic model of $\hat{f}$, fit $\hat{m}_{cu}(\lambda)$ to $\hat{f}(0)$, $\hat{f}'(0)$, and the last two values of $\hat{f}(\lambda)$, and, subject to the same sort of upper and lower limits as before, set λ_k to the value of λ at which $\hat{m}_{cu}(\lambda)$ has its local minimizer (see Figure 6.3.6). The reason for using a cubic is that it can more accurately model situations where f has negative curvature, which are likely when (6.3.3) has failed for two positive values of λ. Furthermore, such a cubic has a unique local minimizer, as illustrated in Figure 6.3.6.

The calculation of λ proceeds as follows. Let λprev and λ2prev be the last two previous values of λ_k. Then the cubic that fits $\hat{f}(0)$, $\hat{f}'(0)$, $\hat{f}(\lambda\text{prev})$, and $\hat{f}(\lambda\text{2prev})$ is

$$\hat{m}_{cu}(\lambda) = a\lambda^3 + b\lambda^2 + \hat{f}'(0)\lambda + \hat{f}(0),$$

where

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\lambda\text{prev} - \lambda\text{2prev}} \times \begin{bmatrix} \dfrac{1}{\lambda\text{prev}^2} & \dfrac{-1}{\lambda\text{2prev}^2} \\ \dfrac{-\lambda\text{2prev}}{\lambda\text{prev}^2} & \dfrac{\lambda\text{prev}}{\lambda\text{2prev}^2} \end{bmatrix} \begin{bmatrix} \hat{f}(\lambda\text{prev}) - \hat{f}(0) - \hat{f}'(0)\lambda\text{prev} \\ \hat{f}(\lambda\text{2prev}) - \hat{f}(0) - \hat{f}'(0)\lambda\text{2prev} \end{bmatrix}.$$

Its local minimizing point $\hat{\lambda}$ is

$$\frac{-b + \sqrt{b^2 - 3a\hat{f}'(0)}}{3a}. \tag{6.3.18}$$

It can be shown that if $\hat{f}(\lambda\text{prev}) \geq \hat{f}(0)$, then $\hat{\lambda} < \frac{2}{3}\lambda\text{prev}$, but this reduction can be achieved in practice and is considered too small (see Example 6.5.1). Therefore the upper bound $u = 0.5$ is imposed, which means that if $\hat{\lambda} > \frac{1}{2}\lambda\text{prev}$, we

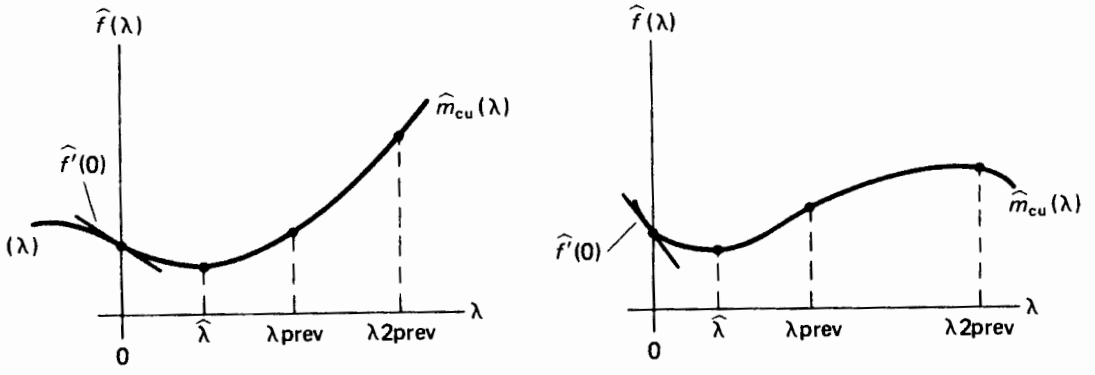


Figure 6.3.6 Cubic backtrack—the two possibilities

set the new $\lambda_k = \frac{1}{2}\lambda\text{prev}$. Also, since $\hat{\lambda}$ can be an arbitrarily small fraction of λprev, the lower bound $l = \frac{1}{10}$ is used again (i.e., if $\lambda < \frac{1}{10}\lambda\text{prev}$, we set the new $\lambda_k = \frac{1}{10}\lambda\text{prev}$). It can be shown that (6.3.18) is never imaginary if α in Algorithm 6.3.5 is less than $\frac{1}{4}$.

There is one important case for which backtracking based on cubic interpolation can be used at every step of Algorithm 6.3.5. Sometimes it is possible to obtain $\nabla f(x_k + \lambda_k p_k)$ at very little additional cost while computing $f(x_k + \lambda_k p_k)$. In this case, $\hat{m}_{cu}(\lambda)$ can be fitted to $\hat{f}(0), \hat{f}'(0), \hat{f}(\lambda\text{prev}), \hat{f}'(\lambda\text{prev})$ at each backtrack. It is a valuable exercise for the student to work out the resulting algorithm in detail.

Algorithm A6.3.1 in the appendix contains our backtracking line-search procedure. It has two additional features. A minimum step length is imposed called *minstep*; it is the quantity used to test for convergence in the upcoming Algorithm A7.2.1. If condition (6.3.3) is not satisfied, but $\| \lambda_k p_k \|_2$ is smaller than *minstep*, then the line search is terminated, since convergence to x_k will be detected at the end of the current iteration anyway. This criterion prevents the line search from looping forever if p_k is not a descent direction. (This sometimes occurs at the final iteration of minimization algorithms, owing to finite-precision errors, especially if the gradient is calculated by finite differences.) Because this condition could also indicate convergence to a false solution, a warning message should be printed. A maximum allowable step length also is declared by the user, because excessively long steps could occur in practice when $p_k = s_k^N = -H_k^{-1} \nabla f(x_k)$ and H_k is nearly singular. The maximum step length is imposed precisely to prevent taking steps that would result in the algorithm's leaving the domain of interest and possibly cause an overflow in the computation of $\hat{f}(1)$.

When we use a line search in conjunction with the secant algorithms of Chapter 9, we will see that we also need to assume that condition (6.3.4) is satisfied at each iteration. Algorithm A6.3.1 mod is the modification of Algorithm A6.3.1 that explicitly enforces this condition. Usually it results in the same steps as Algorithm A6.3.1.

6.4 THE MODEL-TRUST REGION APPROACH

The last section dealt exclusively with the problem of finding an acceptable step length in a given direction of search. The underlying assumptions were that the direction would be the quasi-Newton direction, and that the full quasi-Newton step would always be the first trial step. The resulting backtracking algorithm incorporated these assumptions to attempt global convergence without sacrificing the local convergence properties of the quasi-Newton method, clearly the goal of any global strategy. In this section we seek the

same goal, but we drop the assumption that shortened steps must be in the quasi-Newton direction.

Suppose that the full quasi-Newton step is unsatisfactory. This indicates that our quadratic model does not adequately model f in a region containing the full quasi-Newton step. In line-search algorithms we retain the same *step direction* and choose a shorter *step length*. This new length is determined by building a new one-dimensional quadratic or cubic model, based only on function and gradient information in the quasi-Newton direction. While this strategy is successful, it has the disadvantage that it makes no further use of the n-dimensional quadratic model, including the model Hessian. In this section, when we need to take a shorter step, we first will choose a shorter *step length*, and then use the full n-dimensional quadratic model to choose the *step direction*.

Before we begin to consider ways to choose such directions, it will be useful to make a few preliminary remarks about prespecifying step length. It seems reasonable that after the first iteration we would begin to get at least a rough idea about the length of step we can reasonably expect to make. For example, we might deduce from the length of the step we took at iteration k an upper bound on the length of step likely to be successful at iteration $k+1$. In fact, in Subsection 6.4.3 we will see how to use information about f, gained as the iteration proceeds, to better estimate the length of step likely to be successful. Given this estimate, we might want to start our next iteration with a step of about this length, and not waste possibly expensive function evaluations on longer steps that are unlikely to be successful. Of course, this means that in order to save function evaluations, we might not try a full quasi-Newton step at some iteration when it might be satisfactory. In Subsection 6.4.3 we give some heuristics that are intended to minimize the chances of this occurrence.

Now suppose that we have x_c and some estimate δ_c of the maximum length of a successful step we are likely to be able to take from x_c. This raises the question: how can we best select a step of maximal length δ_c from x_c? A natural answer exists if we return to our idea of a quadratic model. From the beginning we have taken the view that the quasi-Newton step s_c^N is reasonable because it is the step from x_c to the global minimizer of the local quadratic model m_c (if the model Hessian H_c is positive definite). If we add the idea of bounding the maximal step length by $\delta_c > 0$, then the corresponding answer to our question is to try the step s_c that solves

$$\min m_c(x_c + s) = f(x_c) + \nabla f(x_c)^T s + \tfrac{1}{2} s^T H_c s,$$

$$\text{subject to} \quad \|s\|_2 \leq \delta_c. \tag{6.4.1}$$

Problem (6.4.1) is the basis of the "model-trust region" approach to minimization. Its solution is given in Lemma 6.4.1 below. The name comes from viewing δ_c as providing a *region* in which we can *trust* m_c to adequately *model* f.

In the next chapter we will see the merit of using a scaled version of the norm in the step-length bound, but to do so now would only add clutter.

LEMMA 6.4.1 Let $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ be twice continuously differentiable, $H_c \in \mathbb{R}^{n \times n}$ be symmetric and positive definite, and let $\|\cdot\|$ designate the l_2 norm. Then problem 6.4.1 is solved by

$$s(\mu) \triangleq -(H_c + \mu I)^{-1} \nabla f(x_c) \tag{6.4.2}$$

for the unique $\mu \geq 0$ such that $\|s(\mu)\| = \delta_c$, unless $\|s(0)\| \leq \delta_c$, in which case $s(0) = s_c^N$ is the solution. For any $\mu \geq 0$, $s(\mu)$ defines a descent direction for f from x_c.

Proof. (6.4.2) is straightforward from the necessary and sufficient conditions for constrained optimization, but we do not need this generality elsewhere in the book. While reading the proof, the reader may want to refer to Figure 6.4.1. It shows x_c, a positive definite quadratic model m_c with minimum at x^N surrounded by contours at some increasing values of m_c, and a step bound δ_c.

Let us call the solution to (6.4.1) s_*, and let $x_* = x_c + s_*$. Since the Newton point $x_c + s_c^N$ is the global minimizer of m_c, it is clear that if $\|s_c^N\| \leq \delta_c$, then $s_c^N = s_*$. Now consider the case when $x_c + s_c^N$ is outside the step bound as in Figure 6.4.1. Let $\bar{x}$ be any point interior to the constraint region —i.e., $\|\bar{x} - x_c\| < \delta_c$. Then $\nabla m_c(\bar{x}) \neq 0$, so it is possible to decrease m_c from $\bar{x}$ while staying inside the constrained region by

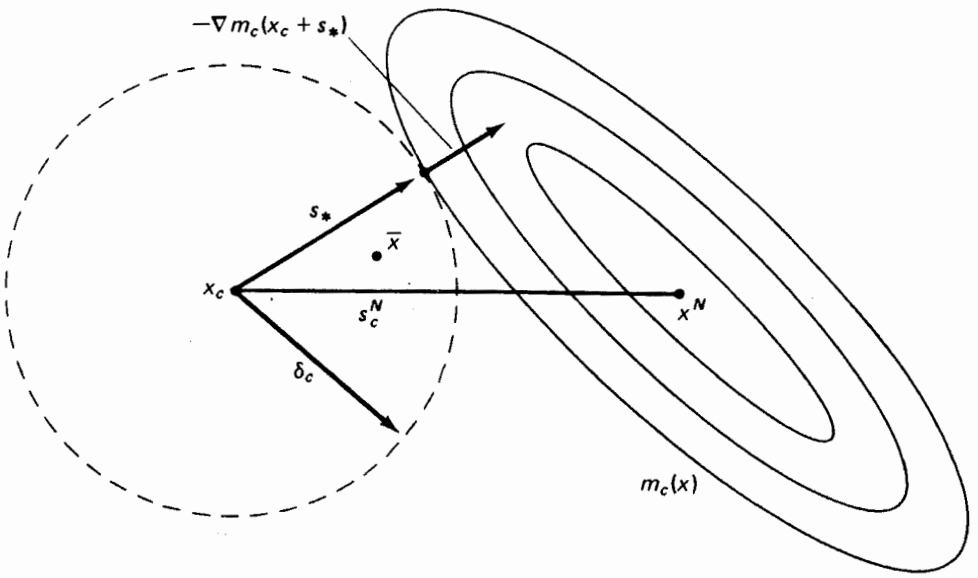


Figure 6.4.1 The solution to (6.4.1)

considering points of the form $\bar{x} - \lambda \nabla m_c(\bar{x})$. This implies that $\bar{s}$ cannot be the solution to (6.4.1), so s_* must satisfy $\|s_*\| = \delta_c$ unless $\|s(0)\| < \delta_c$. Also, since the constrained region in problem (6.4.1) is closed and compact, some s for which $\|s\| = \delta_c$ must be the solution.

Now consider some s such that $\|s\| = \delta_c$. For s to be the solution to (6.4.1), it is necessary that if we move an arbitrarily small distance from $x_c + s$ in any descent direction for m_c, we increase the distance from x_c. A descent direction for m_c from $x_c + s$ is any vector p for which

$$0 > p^T \nabla m_c(x_c + s) = p^T[H_c s + \nabla f(x_c)]. \tag{6.4.3}$$

Similarly a direction $\bar{p}$ from $x_c + s$ increases distance from x_c if and only if

$$\bar{p}^T s > 0. \tag{6.4.4}$$

What we are saying is that for s to solve (6.4.1), any p that satisfies (6.4.3) must also satisfy (6.4.4). Since we know that $\nabla m_c(x_c + s) \neq 0$, this can occur only if $\nabla m_c(x_c + s)$ and $-s$ point in the same direction, or in other words, if for some $\mu > 0$,

$$\mu s = -\nabla m_c(x_c + s) = -(H_c s + \nabla f(x_c))$$

which is just $(H_c + \mu I)s = -\nabla f(x_c)$. Thus $s_* = s(\mu)$ for some $\mu > 0$, when $\|s_c^N\| > \delta_c$. Since $\mu \geq 0$ and H_c is symmetric and positive definite, $(H_c + \mu I)$ is symmetric and positive definite, so $s(\mu)$ is a descent direction for f from x_c. In order to finish the proof, we need to show that s_* is unique, which is implied by the stronger result that if $\mu_1 > \mu_2 \geq 0$, then $\|s(\mu_1)\| < \|s(\mu_2)\|$. This shows that $\|s(\mu)\| = \delta_c$ has a unique solution that must be the solution to (6.4.1). The proof is straightforward, because if $\mu \geq 0$ and

$$\eta(\mu) \triangleq \|s(\mu)\|_2 = \|(H_c + \mu I)^{-1} \nabla f(x_c)\|,$$

then

$$\eta'(\mu) = \frac{-\nabla f(x_c)^T(\mu I + H_c)^{-3} \nabla f(x_c)}{\|s(\mu)\|} = \frac{-s(\mu)^T(H_c + \mu I)^{-1}s(\mu)}{\|s(\mu)\|}$$

and so $\eta'(\mu) < 0$ as long as $\nabla f(x_c) \neq 0$. Thus η is a monotonically decreasing function of μ. □

The trouble with using Lemma 6.4.1 as the basis of a step in a minimization algorithm is that there is no finite method of determining the $\mu_c > 0$ such that $\|s(\mu_c)\|_2 = \delta_c$, when $\delta_c < \|H_c^{-1} \nabla f(x_c)\|_2$. Therefore, in the next two subsections we will describe two computational methods for approximately solving problem (6.4.1). The first, the locally constrained optimal (or "hook") step, finds a μ_c such that $\|s(\mu_c)\|_2 \cong \delta_c$, and takes $x_+ = x_c + s(\mu_c)$. The second, the dogleg step, makes a piecewise linear approximation to the curve $s(\mu)$, and takes x_+ as the point on this approximation for which $\|x_+ - x_c\| = \delta_c$.

There is no guarantee that the x_+ that approximately or exactly solves (6.4.1) will be an acceptable next point, although we hope it will be if δ_c is a good step bound. Therefore, a complete step of a trust-region algorithm will have the following form:

ALGORITHM 6.4.2 A GLOBAL STEP BY THE MODEL-TRUST REGION APPROACH

Given $f: \mathbb{R}^n \longrightarrow \mathbb{R}$, $\delta_c > 0$, $x_c \in \mathbb{R}^n$, $H_c \in \mathbb{R}^{n \times n}$ symmetric and positive definite:

repeat
(1) $s_c :=$ approximate solution to (6.4.1),
$x_+ := x_c + s_c$,
(2) decide whether x_+ is acceptable, and calculate a new value of δ_c

until x_+ is an acceptable next point;

$\delta_+ := \delta_c$

To complete the model-trust region algorithm, we discuss the above Step 2 in Subsection 6.4.3.

Finally, some observations are in order. First, Gay (1981) shows that even if H_c has negative eigenvalues, the solution to (6.4.1) is still an s_* satisfying $(H_c + \mu I)s_* = -\nabla f(x_c)$ for some $\mu > 0$ such that $H_c + \mu I$ is at least positive semidefinite. This leads to another justification for our strategy from Section 5.5 of perturbing the model Hessian to a positive definite $H_c = \nabla^2 f(x_c) + \mu I$ when $\nabla^2 f(x_c)$ is not positive definite; the resultant quasi-Newton step $-H_c^{-1}\nabla f(x_c)$ is to the minimizer of the original (non positive definite) quadratic model in some spherical region around x_c. For the remainder of Section 6.4 we will assume that H_c is positive definite.

Second, it is important to note that $s(\mu)$ runs smoothly from the quasi-Newton step $-H_c^{-1}\nabla f(x_c)$, when $\mu = 0$, to $s(\mu) \cong -(1/\mu)\nabla f(x_c)$, when μ gets large. Thus when δ_c is very small, the solution to (6.4.1) is a step of length δ_c in approximately the steepest-descent direction. Figure 6.4.2 traces the curve $s(\mu)$, $0 \le \mu < \infty$, for the same quadratic model as in Figure 6.4.1. We note that in general when $n > 2$, this curve does not lie in the subspace spanned by $\nabla f(x_c)$ and $H_c^{-1}\nabla f(x_c)$ (see Exercise 6.10).

Fletcher (1980) calls these algorithms *restricted step methods*. This term emphasizes the procedure to be covered in Subsection 6.4.3 for updating δ_c, which could also be used in line-search algorithms. Some important groundwork for the algorithms of this section was provided by Levenberg (1944), Marquardt (1963), and Goldfeldt, Quandt, and Trotter (1966). More recent references are cited in the following pages.

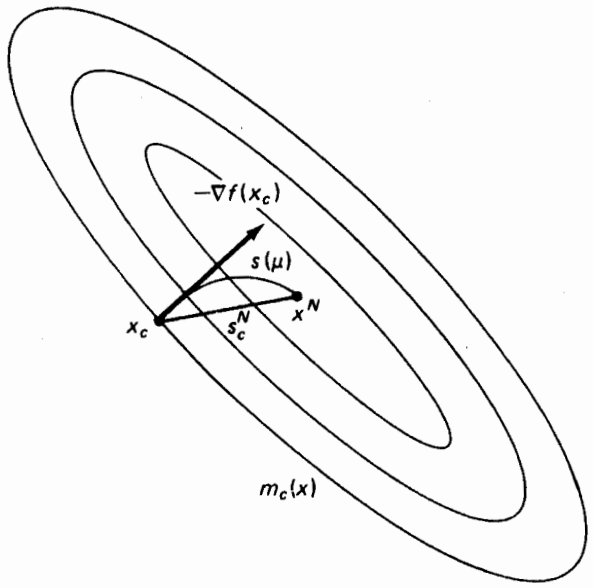


Figure 6.4.2 The curve $s(\mu)$

6.4.1 THE LOCALLY CONSTRAINED OPTIMAL ("HOOK") STEP

Our first approach for calculating the step in model-trust region Algorithm 6.4.2, when $\| s(0) \|_2 > \delta_c$, is to find an $s(\mu) \triangleq -(H_c + \mu I)^{-1} \nabla f(x_c)$ such that $\| s(\mu) \|_2 \cong \delta_c$, and then make a trial step to $x_+ := x_c + s(\mu)$. In this section we discuss an algorithm for finding an approximate solution μ_c to the scalar equation

$$\Phi(\mu) \triangleq \| s(\mu) \|_2 - \delta_c = 0. \tag{6.4.5}$$

In practice we will not require a very exact solution at all, but we defer that consideration to the end of this subsection.

An application of the ideas of Chapter 2 would cause us to use Newton's method to solve (6.4.5). Although this would work, it can be shown that for all $\mu \geq 0$, $\Phi'(\mu) < 0$ and $\Phi''(\mu) > 0$, so Newton's method will always underestimate the exact solution μ_* (see Figure 6.4.3). Therefore, we will construct a different method geared specifically to (6.4.5).

The proper idea to bring from Chapter 2 is the device of a local model of the problem to be solved. In this case, the one-dimensional version

$$\Phi(\mu) = | -(\mu + h_c)^{-1} g_c | - \delta_c$$

suggests a local model of the form

$$m_c(\mu) = \frac{\alpha_c}{\beta_c + \mu} - \delta_c$$

with two free parameters α, β. Before we discuss α, β, notice that we have slipped into a new notation for dealing with the iteration on μ, which comes

from the model at
purpose is to find t
values of the quanti
and normal type fo
the outer iteration :
obtain μ_c as the ini
tion of (6.4.5); and v

The model m_c
derivation of Newt
two conditions:

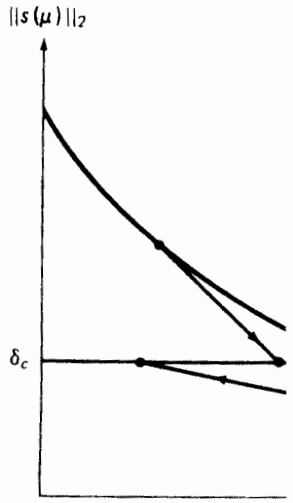
$$\mathbf{m_c}(\mu_c) = \bar{}$$
$$\boldsymbol{\beta}$$

and

$$\mathbf{m_c'}(\mu_c) = \cdot$$

This gives

Notice that the com
so it will require $O(n$
the computation of
relatively cheap.



Figure

from the model above, takes place inside the main x-iteration, and whose purpose is to find the μ_c satisfying (6.4.5). We will use boldface for the current values of the quantities $\boldsymbol{\alpha}_c$, $\boldsymbol{\beta}_c$, $\boldsymbol{\mu}_c$ that are changing in the inner iteration on μ, and normal type for the current values of δ_c, x_c, $\nabla f(x_c)$, H_c that come from the outer iteration and are unchanged during the solution of (6.4.5). Thus, we obtain μ_c as the inner iteration's last approximation $\boldsymbol{\mu}_c$ to $\boldsymbol{\mu}_*$, the exact solution of (6.4.5); and we use x_c and μ_c to define $x_+ = x_c + s(\mu_c)$.

The model $\boldsymbol{m}_c(\boldsymbol{\mu})$ has two free parameters, $\boldsymbol{\alpha}_c$ and $\boldsymbol{\beta}_c$, and so as in the derivation of Newton's method, it is reasonable to choose them to satisfy the two conditions:

$$\mathbf{m}_c(\boldsymbol{\mu}_c) = \frac{\boldsymbol{\alpha}_c}{\boldsymbol{\beta}_c + \boldsymbol{\mu}_c} - \delta_c = \Phi(\boldsymbol{\mu}_c) = \| s(\boldsymbol{\mu}_c) \|_2 - \delta_c$$

and

$$\mathbf{m}_c'(\boldsymbol{\mu}_c) = -\frac{\boldsymbol{\alpha}_c}{(\boldsymbol{\beta}_c + \boldsymbol{\mu}_c)^2} = \Phi'(\boldsymbol{\mu}_c) = -\frac{s(\boldsymbol{\mu}_c)^T (H_c + \boldsymbol{\mu}_c I)^{-1} s(\boldsymbol{\mu}_c)}{\| s(\boldsymbol{\mu}_c) \|_2}$$

This gives

$$\boldsymbol{\alpha}_c = -\frac{(\Phi(\boldsymbol{\mu}_c) + \delta_c)^2}{\Phi'(\boldsymbol{\mu}_c)}, \tag{6.4.6}$$

$$\boldsymbol{\beta}_c = -\frac{(\Phi(\boldsymbol{\mu}_c) + \delta_c)}{\Phi'(\boldsymbol{\mu}_c)} - \boldsymbol{\mu}_c. \tag{6.4.7}$$

Notice that the computation of $s(\boldsymbol{\mu}_c)$ requires a factorization of $H_c + \boldsymbol{\mu}_c I$, and so it will require $O(n^3)$ arithmetic operations. Once we have this factorization, the computation of $(H_c + \boldsymbol{\mu}_c I)^{-1} s(\boldsymbol{\mu}_c)$ costs only $O(n^2)$ operations, so $\Phi'(\boldsymbol{\mu}_c)$ is relatively cheap.

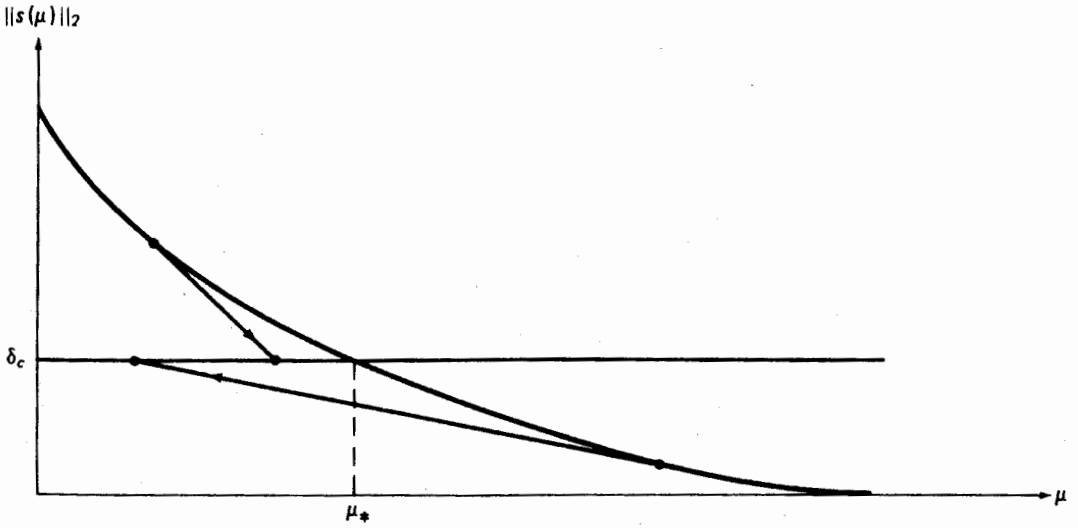


Figure 6.4.3 Newton's method for finding $\Phi(\mu) = 0$

Now that we have our model $\mathbf{m}_c(\mu)$, we naturally choose μ_+ such that $\mathbf{m}_c(\mu_+) = 0$—that is

$$\mu_+ = \frac{\alpha_c}{\delta_c} - \beta_c.$$

The reader can verify that substituting (6.4.6), (6.4.7), and (6.4.5) into the above gives

$$\begin{aligned}\mu_+ &= \mu_c - \left[\frac{\Phi(\mu_c) + \delta_c}{\delta_c}\right]\left[\frac{\Phi(\mu_c)}{\Phi'(\mu_c)}\right] \\ &= \mu_c - \frac{\| s(\mu_c)\|}{\delta_c}\left[\frac{\Phi(\mu_c)}{\Phi'(\mu_c)}\right] \qquad (6.4.8)\end{aligned}$$

as our iteration for solving $\Phi(\mu) = 0$. The form of (6.4.8) shows that when $\mu_c < \mu_*$, we are taking a larger correction than Newton's method would give, but that when $\mu_c > \mu_*$, the step is smaller than the Newton step. As $\mu_c \longrightarrow \mu_*$, (6.4.8) becomes more and more like Newton's method; in fact, (6.4.8) is locally q-quadratically convergent to μ_*.

Several issues remain in transforming (6.4.8) into a computational algorithm. The following discussion, and our algorithm, is based on Hebden (1973) and Moré (1977). Another comprehensive reference is Gander (1978). Readers not interested in implementation details may wish to skip the next three paragraphs.

The first consideration is a starting value for μ in solving (6.4.5). Reinsch (1971) has shown that if (6.4.8) is started with $\mu_0 = 0$, then the μ iteration converges monotonically up to μ_*, but we would like a closer start because each iteration of (6.4.8) involves solving a linear system. Moré's implementation uses the approximate solution μ_- to the last instance of (6.4.5) to generate an initial guess for the current instance. The rule Moré uses is simple: if the current step bound δ_c is p times the last value of the step bound δ_-, then $\mu_0 = \mu_-/p$ is used to start (6.4.8). We prefer a different strategy. Remember where we are in the iteration: we have just made an x-step $s(\mu_-)$ from x_- to x_c, and we now want x_+; we have obtained δ_c from δ_-. Before we get H_c, we still have $(H_- + \mu_- I)$ in factored form, and so it costs only $O(n^2)$ to compute

$$\Phi \triangleq \| s(\mu_-)\| - \delta_c \quad \text{and} \quad \Phi' \triangleq \frac{s(\mu_-)^T(H_- + \mu_- I)^{-1} s(\mu_-)}{\| s(\mu_-)\|}.$$

In analogy with (6.4.8), we use

$$\mu_0 = \mu_- - \frac{\| s(\mu_-)\|}{\delta_c}\frac{\Phi}{\Phi'}, \qquad (6.4.9)$$

the value of μ gotten from μ_-, the previous model, and the new trust radius δ_c. If $\delta_c = \delta_-$, then μ_0 is exactly the value we would have gotten had we taken one more μ-iteration on the previous model. On the other hand, if the previous iteration took the Newton step, then we find μ_0 by a different technique mentioned later.

Another computational detail, important to the performance of the algorithm for solving (6.4.5), is the generation and updating of lower and upper bounds u_+ and l_+ on μ_+. These bounds are used with (6.4.8) in the same way that bounds were used to safeguard the backtrack steps in Algorithm 6.3.5; that is, we restrict μ_+ to be in $[l_+, u_+]$. Since the Newton iteration applied to (6.4.5) always undershoots μ_*, we take $l_0 = -\Phi(0)/\Phi'(0)$. We then calculate

$$\mu_+^N = \mu_c - \Phi(\mu_c)/\Phi'(\mu_c)$$

along with each calculation of (6.4.8), and update the lower bound to $l_+ = \max\{l_c, \mu_+^N\}$, where l_c is the current lower bound. Also, since

$$\delta_c = \|(H_c + \mu_* I)^{-1} \nabla f(x_c)\|_2 < \frac{\|\nabla f(x_c)\|_2}{\mu_*}$$

because H_c is positive definite and $\mu_* > 0$, we take $\|\nabla f(x_c)\|_2/\delta_c$ as our initial upper bound u_0 on μ_*. Then, at each iteration, if $\Phi(\mu_c) < 0$ we update the upper bound to $u_+ = \min\{u_c, \mu_c\}$, where u_c is the current upper bound.

If, at any iteration, μ_+ is not in $[l_+, u_+]$, we follow Moré in choosing μ_+ by

$$\mu_+ = \max\{(l_+ \cdot u_+)^{1/2}, 10^{-3}u_+\}, \tag{6.4.10}$$

the second term being a safeguard against near-zero values of l_+. In practice, these bounds are invoked most frequently in calculating μ_0. In particular, (6.4.10) is used to define μ_0 whenever (6.4.9) cannot be used because the previous iteration used the Newton step.

Finally, we do not solve (6.4.5) to any great accuracy, settling instead for

$$\|s(\mu)\|_2 \in \left[\frac{3\delta_c}{4}, \frac{3\delta_c}{2}\right]$$

The reason is that, as will be seen in Subsection 6.4.3, the trust region is never increased or decreased by a factor smaller than 2. So if the current trust radius is δ_c, the previous one was either greater than or equal to $2\delta_c$, or less than or equal to $\delta_c/2$. Thus we consider the actual value of δ_c to be rather arbitrary within the range $[3\delta_c/4, 3\delta_c/2]$, which splits the difference in either direction, and it seems reasonable to allow $\|s(\mu)\|_2$ to have any value in this range. Some other implementations, for example Moré (1977), require $\|s(\mu)\|_2 \in [0.9\delta_c, 1.1\delta_c]$, and it is not clear whether either is better. Experience shows that either choice is satisfied in practice with an average of between 1 and 2 values of μ per x-iteration.

We have now completed our discussion of an algorithm for approximately solving (6.4.5); a simple example is given below.

EXAMPLE 6.4.3 Let $f(x_1, x_2) = x_1^4 + x_1^2 + x_2^2$, $x_c = (1, 1)^T$, $H_c = \nabla^2 f(x_c)$, $\delta_c = \frac{1}{2}$, $\mu_- = 0$. Then

$$\nabla f(x_c) = (6, 2)^T,$$

$$H_c = \begin{bmatrix} 14 & 0 \\ 0 & 2 \end{bmatrix},$$

$$s_c^N = -H_c^{-1} \nabla f(x_c) = (-\tfrac{3}{7}, -1)^T,$$

$$\| s_c^N \|_2 \cong 1.088.$$

Since $\| s_c^N \|_2 > \frac{3}{2}\delta_c$, the Newton step is too long, and we seek some $\mu > 0$ such that

$$\| (H_c + \mu I)^{-1} \nabla f(x_c) \|_2 \in \left[\frac{3\delta_c}{4}, \frac{3\delta_c}{2} \right] = [0.375, 0.75]$$

(see Figure 6.4.4). Since $\mu_- = 0$, we calculate

$$\mathbf{l}_0 = 0 - \frac{\Phi(0)}{\Phi'(0)} \cong \frac{0.588}{0.472} \cong 1.25,$$

$$\mathbf{u}_0 = \frac{\| \nabla f(x_c) \|_2}{\delta_c} \cong \frac{6.325}{0.5} \cong 12.6$$

$$\boldsymbol{\mu}_0 = (\mathbf{l}_0 \cdot \mathbf{u}_0)^{1/2} \cong 3.97.$$

Next we calculate $s(\boldsymbol{\mu}_0) = -(H_c + \boldsymbol{\mu}_0 I)^{-1} \nabla f(x_c) \cong (-0.334, -0.335)^T$. Since

$$\| s(\boldsymbol{\mu}_0) \|_2 \cong 0.473 \in [0.375, 0.75],$$

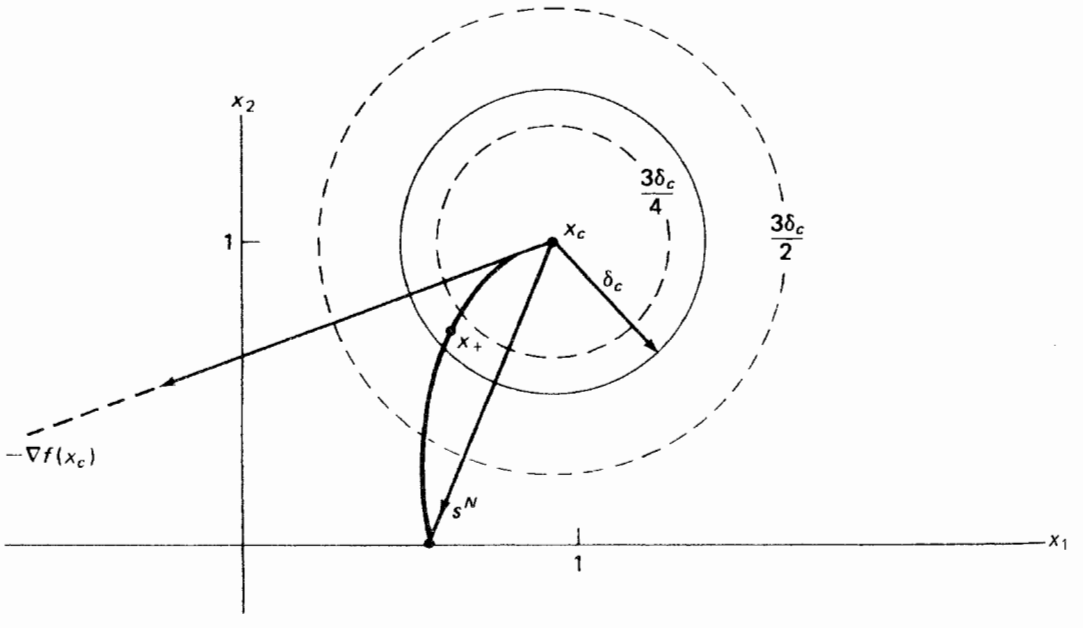


Figure 6.4.4 "Hook" step for Example 6.4.3

we take $\mu_c = \mu_0$ and $x_+ = x_c + s(\mu_c) \cong (0.666, 0.665)^T$ as our approximate solution to (6.4.1). Note that in this case we have not used iteration (6.4.8). For illustration, the reader can verify that one application of (6.4.8) would result in

$$\mu_1 = \mu_0 - \frac{\Phi(\mu_0)}{\Phi'(\mu_0)} \cdot \frac{\| s(\mu_0) \|_2}{\delta_c} = 3.97 - \frac{-0.0271}{-0.0528} \cdot \frac{+0.473}{+0.5} = 3.49$$

and that $s(\mu_1) = (-0.343, -0.365)^T$, $\| s(\mu_1) \|_2 = 0.5006$. Incidentally, (6.4.5) is solved exactly by $\mu_* \cong 3.496$.

Our algorithm for computing the locally constrained optimal ("hook") step by the techniques described in this section is given in Algorithm A6.4.2 in the appendix. It is preceded by the driver for a complete global step using the locally constrained optimal approach in Algorithm A6.4.1. The complete step involves selecting a new point x_+ by Algorithm A6.4.2, checking if x_+ is satisfactory and updating the trust radius (Algorithm A6.4.5), and repeating this process if necessary. All the algorithms use a diagonal scaling matrix on the variable space that will be discussed in Section 7.1.

Some recent research [Gay (1981), Sorensen (1982)] has centered on approximately solving the locally constrained model problem (6.4.1) by the techniques of this section expanded to cover the case when H_c is not positive definite. These techniques may become an important addition to the algorithms of this section.

6.4.2 THE DOUBLE DOGLEG STEP

The other implementation of the model-trust region approach that we discuss is a modification of the trust region algorithm introduced by Powell (1970a). It also finds an approximate solution to problem (6.4.1). However, rather than finding a point $x_+ = x_c + s(\mu_c)$ on the $s(\mu)$ curve such that $\| x_+ - x_c \|_2 \cong \delta_c$, it approximates this curve by a piecewise linear function connecting the "Cauchy point" C.P., the minimizer of the quadratic model m_c in the steepest-descent direction, to the Newton direction for m_c, as indicated in Figure 6.4.5. Then it chooses x_+ to be the point on this polygonal arc such that $\| x_+ - x_c \|_2 = \delta_c$, unless $\| H_c^{-1} \nabla f(x_c) \|_2 < \delta_c$, in which case x_+ is the Newton point. This strategy can alternatively be viewed as a simple strategy for looking in the steepest-descent direction when δ_c is small, and more and more toward the quasi-Newton direction as δ_c increases.

The specific way of choosing the double dogleg curve makes it have two important properties. First, as one proceeds along the piecewise linear curve from x_c to C.P. to $\hat{N}$ to x_+^N, the distance from x_c increases monotonically. Thus for any $\delta \leq \| H_c^{-1} \nabla f(x_c) \|$, there is a unique point x_+ on the curve such that $\| x_+ - x_c \|_2 = \delta$. This just makes the process well defined. Second, the value of the quadratic model $m_c(x_c + s)$ decreases monotonically as s goes

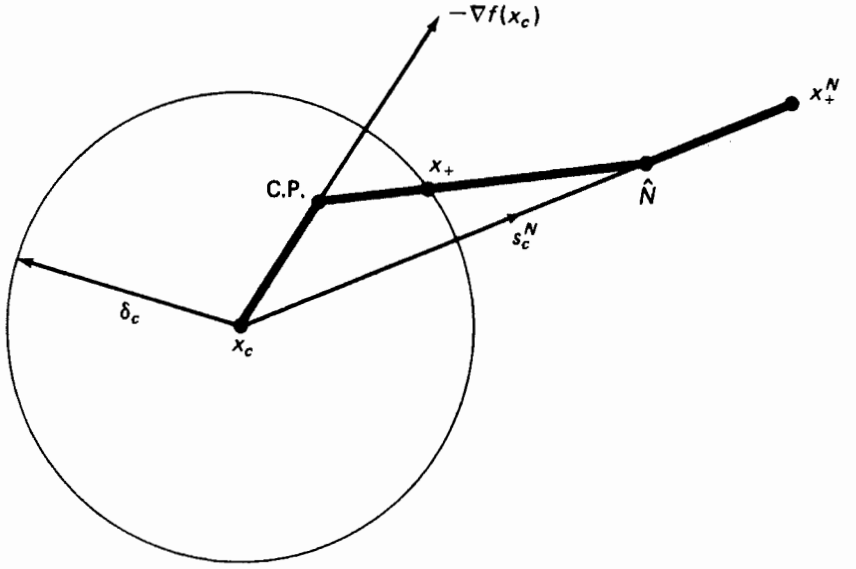


Figure 6.4.5 The double dogleg curve, $x_c \to \text{C.P.} \to \hat{N} \to x_+^N$

along the curve from x_c to C.P. to $\hat{N}$ to x_+^N. This makes the process reasonable.

Point C.P. in Figure 6.4.5 is found by solving

$$\min_{\lambda \in \mathbb{R}} m_c(x - \lambda \nabla f(x_c)) = f(x_c) - \lambda \|\nabla f(x_c)\|_2^2 + \tfrac{1}{2}\lambda^2 \nabla f(x_c)^T H_c \nabla f(x_c),$$

which has the unique solution

$$\lambda_* = \frac{\|\nabla f(x_c)\|_2^2}{\nabla f(x_c)^T H_c \nabla f(x_c)}.$$

Therefore,

$$\text{C.P.} = x_c - \lambda_* \nabla f(x_c),$$

and if

$$\delta_c \le \lambda_* \|\nabla f(x_c)\|_2 = \|\nabla f(x_c)\|_2^3 / \nabla f(x_c)^T H_c \nabla f(x_c),$$

the algorithm takes a step of length δ_c in the steepest-descent direction:

$$x_+ = x_c - \frac{\delta_c}{\|\nabla f(x_c)\|_2} \nabla f(x_c).$$

In order for the double dogleg curve to satisfy the first property stated above, it must be shown that the Cauchy point C.P. is no farther from x_c than the Newton point x_+^N. Let

$$s^{\text{C.P.}} = -\lambda_* \nabla f(x_c),\ s^N = -H_c^{-1} \nabla f(x_c).$$

Then

$$\|s^{\text{C.P.}}\|_2 = \frac{\|\nabla f(x_c)\|_2^3}{\nabla f(x_c)^T H_c\, \nabla f(x_c)}$$

$$\leq \frac{\|\nabla f(x_c)\|_2^3}{\nabla f(x_c)^T H_c\, \nabla f(x_c)} \frac{\|\nabla f(x_c)\|_2 \, \|H_c^{-1}\, \nabla f(x_c)\|_2}{\nabla f(x_c)^T H_c^{-1}\, \nabla f(x_c)}$$

$$= \frac{\|\nabla f(x_c)\|_2^4}{(\nabla f(x_c)^T H_c\, \nabla f(x_c))(\nabla f(x_c)^T H_c^{-1} \nabla f(x_c))} \|s^N\|_2 \triangleq \gamma \|s^N\|_2 . \qquad (6.4.11)$$

It is an exercise to prove that $\gamma \leq 1$ for any positive definite H_c, with $\gamma = 1$ only if $S^{\text{C.P.}} = S^N$, a case which we exclude for the remainder of this section. Thus,

$$\|s^{\text{C.P.}}\|_2 \leq \gamma \|s^N\|_2 \leq \|s^N\|_2 .$$

The point $\hat{N}$ on the double dogleg curve is now chosen to have the form

$$\hat{N} = x_c - \eta H_c^{-1}\, \nabla f(x_c)$$

for some η such that

$$\gamma \leq \eta \leq 1$$

and

$$m_c(x) \text{ decreases monotonically along the line from C.P. to } \hat{N}. \qquad (6.4.12)$$

Since we know that $m_c(x)$ decreases monotonically from x_c to C.P. and from $\hat{N}$ to x_+^N, (6.4.12) will guarantee that $m_c(x)$ decreases monotonically along the entire double dogleg curve.

To satisfy (6.4.12), η must be chosen so that the directional derivative along the line connecting C.P. and $\hat{N}$ is negative at every point on that line segment. Parametrize this line segment by

$$x_+(\lambda) = x_c + s^{\text{C.P.}} + \lambda(\eta s^N - s^{\text{C.P.}}) \qquad 0 \leq \lambda \leq 1.$$

The directional derivative of m_c along this line at $x_+(\lambda)$ is

$$\nabla m_c(x_+(\lambda))^T(\eta s^N - s^{\text{C.P.}}) = [\nabla f(x_c) + H_c(s^{\text{C.P.}} + \lambda(\eta s^N - s^{\text{C.P.}}))]^T(\eta s^N - s^{\text{C.P.}})$$

$$= (\nabla f(x_c) + H_c\, s^{\text{C.P.}})^T(\eta s^N - s^{\text{C.P.}})$$

$$+ \lambda(\eta s^N - s^{\text{C.P.}})^T H_c(\eta s^N - s^{\text{C.P.}}). \qquad (6.4.13)$$

Since H_c is positive definite, the right-hand side of (6.4.13) is a monotonically increasing function of λ. Therefore, we need only require that (6.4.13) be negative for $\lambda = 1$ to make it negative for $0 \leq \lambda \leq 1$. Some cancellation and substi-

tution shows that this condition is equivalent to

$$0 > (1-\eta)(\nabla f(x_c)^T(\eta s^N - s^{C.P.})) = (1-\eta)(\gamma-\eta)(\nabla f(x_c)^T H_c^{-1}\,\nabla f(x_c)),$$

which is satisfied for any $\eta \in (\gamma, 1)$.

Thus $\hat{N}$ can be chosen as any point in the Newton direction $x_c + \eta s^N$ for which η is between 1 and γ given by (6.4.11). Powell's original choice was $\eta = 1$, giving the single dogleg curve. Computational testing, however, has shown that an earlier bias to the Newton direction seems to improve the performance of the algorithm. Therefore, Dennis and Mei (1979) suggest the choice $\eta = 0.8\gamma + 0.2$, leading to a double dogleg curve as in Figure 6.4.5.

The choices of C.P. and $\hat{N}$ completely specify the double dogleg curve. The selection of the x_+ on the curve such that $\|x_+ - x_c\|_2 = \delta_c$ is then a straightforward inexpensive algebraic problem, as illustrated in Example 6.4.4 below. Notice that the entire algorithm costs only $O(n^2)$ arithmetic operations after s_c^N has been calculated.

EXAMPLE 6.4.4 Let $f(x)$, x_c, H_c be given by Example 6.4.3, and let $\delta_c = 0.75$. Recall that

$$\nabla f(x_c) = (6, 2)^T, \qquad H_c = \begin{bmatrix} 14 & 0 \\ 0 & 2 \end{bmatrix}, \qquad s_c^N = (-\tfrac{3}{7}, -1)^T.$$

Since $\|s_c^N\|_2 = 1.088 > \delta_c$, the double dogleg algorithm first calculates the step to the Cauchy point. The reader can verify that it is given by

$$s^{C.P.} = -\left(\frac{40}{512}\right)\begin{pmatrix} 6 \\ 2 \end{pmatrix} \cong \begin{pmatrix} -0.469 \\ -0.156 \end{pmatrix}.$$

Since $\|s^{C.P.}\|_2 \cong 0.494 < \delta_c$, the algorithm next calculates the step to the point $\hat{N}$. The reader can verify that

$$\gamma = \frac{(40)^2}{(512)(\frac{32}{7})} \cong 0.684,$$

$$\eta = 0.8\gamma + 0.2 \cong 0.747,$$

$$s^{\hat{N}} \triangleq \eta s_c^N \cong \begin{pmatrix} -0.320 \\ -0.747 \end{pmatrix}.$$

Since $\|s^{\hat{N}}\|_2 \cong 0.813 > \delta_c$, the double dogleg step must be along the line connecting C.P. and $\hat{N}$—that is, $s_c = s^{C.P.} + \lambda(s^{\hat{N}} - s^{C.P.})$ for the $\lambda \in (0, 1)$ for which $\|s_c\|_2 = \delta_c$. λ is calculated by solving for the positive root of the quadratic equation

$$\|s^{C.P.} + \lambda(s^{\hat{N}} - s^{C.P.})\|_2^2 = \delta_c^2,$$

Fi

which is $\lambda \cong 0.86$

The entire calcula

Our algorit
given in Algorith
that the points s
those selected by
However, there is
A6.4.4 are consid
dogleg strategy a
and derivative eva

Algorithm A
double dogleg alg
the diagonal scalin

6.4.3 UPDATIN

To complete
whether the point
6.4.2, is a satisfacto
the trust region an
region. If x_+ is sa
be increased, decre
The basis for these

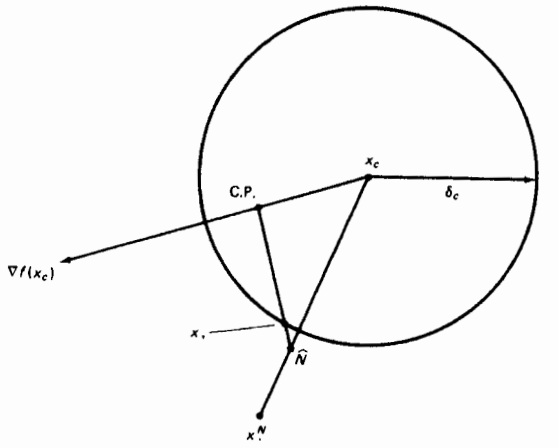


Figure 6.4.6 The double dogleg step of Example 6.4.4

which is $\lambda \cong 0.867$. Thus,

$$s_c = s^{\text{C.P.}} + 0.867(s^{\hat{N}} - s^{\text{C.P.}}) \cong \begin{pmatrix} -0.340 \\ -0.669 \end{pmatrix},$$

$$x_+ = x_c + s_c \cong \begin{pmatrix} 0.660 \\ 0.331 \end{pmatrix}.$$

The entire calculation is shown in Figure 6.4.6.

Our algorithm for selecting a point by the double dogleg strategy is given in Algorithm A6.4.4 in the appendix. Computational experience shows that the points selected by this strategy are, at worst, marginally inferior to those selected by the locally constrained optimal step of Algorithm A6.4.2. However, there is a trade-off: the complexity and running time of Algorithm A6.4.4 are considerably less than those of Algorithm A6.4.2. This makes the dogleg strategy attractive, especially for problems where the cost of function and derivative evaluation is not high.

Algorithm A6.4.3 contains the driver for a complete global step using the double dogleg algorithm to find candidate points x_+. Both algorithms contain the diagonal scaling of the variable space that is described in Section 7.1.

6.4.3 *UPDATING THE TRUST REGION*

To complete the global step given in Algorithm 6.4.2, one needs to decide whether the point x_+, found by using the techniques of Subsection 6.4.1 or 6.4.2, is a satisfactory next iterate. If x_+ is unacceptable, one reduces the size of the trust region and minimizes the same quadratic model on the smaller trust region. If x_+ is satisfactory, one must decide whether the trust region should be increased, decreased, or kept the same for the next step (of Algorithm 6.1.1). The basis for these decisions is discussed below.

The condition for accepting x_+ is the one developed in Section 6.3,

$$f(x_+) \leq f(x_c) + \alpha g_c^T(x_+ - x_c), \tag{6.4.14}$$

where $g_c = \nabla f(x_c)$ or an approximation to it, and α is a constant in $(0, \frac{1}{2})$. In our algorithm we again choose $\alpha = 10^{-4}$, so that (6.4.14) is hardly more stringent than $f(x_+) < f(x_c)$. If x_+ does not satisfy (6.4.14), we reduce the trust region by a factor between $\frac{1}{10}$ and $\frac{1}{2}$ and return to the approximate solution of the locally constrained minimization problem by the locally constrained optimal step or double dogleg method. The reduction factor is determined by the same quadratic backtrack strategy used to decrease the line-search parameter in Algorithm A6.3.1. We model $f(x_c + \lambda(x_+ - x_c))$ by the quadratic $m_q(\lambda)$ that fits $f(x_c)$, $f(x_+)$, and the directional derivative $g_c^T(x_+ - x_c)$ of f at x_c in the direction $x_+ - x_c$. We then let the new trust radius δ_+ extend to the minimizer of this model, which occurs at

$$\lambda_* = \frac{-g_c^T(x_+ - x_c)}{2[f(x_+) - f(x_c) - g_c^T(x_+ - x_c)]}$$

(see Figure 6.4.7). Thus, $\delta_+ = \lambda_* \| x_+ - x_c \|_2$. If

$$\lambda_* \| x_+ - x_c \|_2 \notin [\tfrac{1}{10}\delta_c, \tfrac{1}{2}\delta_c],$$

we instead set δ_+ to the closer endpoint of this interval for the same reasons as in the line-search algorithm. Note that $\| x_+ - x_c \|_2 = \delta_c$ if x_+ is selected by the double dogleg strategy of Subsection 6.4.2, but that we only know that $\| x_+ - x_c \|_2 \in [\frac{3}{4}\delta_c, \frac{3}{2}\delta_c]$ if x_+ is selected by the locally constrained optimal strategy of Subsection 6.4.1.

Now suppose we have just found an x_+ that satisfies (6.4.14). If x_+ is a full Newton step from x_c, then we make the step, update δ, form the new model, and go to the next iteration. However, if $x_+ - x_c$ isn't the Newton step, we first consider whether we should try a larger step from x_c, using the current

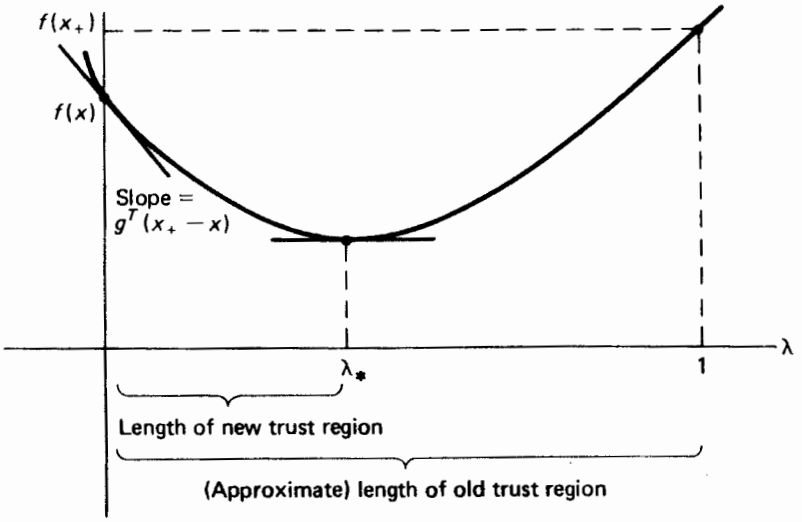


Figure 6.4.7 Reducing the trust region when x_+ is unacceptable.

model. The reason this may be worthwhile is that we may thereby avoid having to evaluate the gradient (and Hessian) at x_+, which is often the dominant cost in expensive problems. The reason a longer step may be possible is that the trust region may have become small during the course of the algorithm and may now need to be increased. This occurs when we leave a region where the step bound had to be small because the function was not well represented by any quadratic, and enter a region where the function is more nicely behaved.

To decide whether to attempt a larger step from x_c, we compare the actual reduction $\Delta f \triangleq f(x_+) - f(x_c)$ to the predicted reduction $\Delta f_{\text{pred}} \triangleq m_c(x_+) - f(x_c)$ and accept x_+ as the next iterate unless either the agreement is so good—i.e., $|\Delta f_{\text{pred}} - \Delta f| \leq 0.1\,|\Delta f|$—that we suspect δ_c is an underestimate of the radius in which m_c adequately represents f, or the actual reduction in f is so large that the presence of negative curvature, and thus a continuing rapid decrease in $f(x)$, is implied—i.e., $f(x_+) \leq f(x_c) + \nabla f(x_c)^T(x_+ - x_c)$. In both these cases we save x_+ and $f(x_+)$, but rather than move directly to x_+, we first double δ_c and compute a new x_+ using our current model. If (6.4.14) isn't satisfied for the new x_+, we drop back to the last good step we computed, but if it is, we consider doubling again. In practice we may save a significant number of gradient calls in this way.

An interesting situation in which the step bound must contract and then expand is when the algorithm passes close by a point that looks like a minimizer. The Newton steps shorten; the algorithm takes these Newton steps and behaves as though it were converging. Then the algorithm discovers a way out, the Newton steps lengthen, and the algorithm moves away. Such behavior is desirable, since a perturbed problem might indeed have a real minimizer at this point of distraction. We want to be able to increase δ_c rapidly in such a case. In one example, we saw six of these internal doublings of the step bound after a point of distraction.

Now suppose we are content with x_+ as our next iterate, and so we need to update δ_c to δ_+. We allow three alternatives: doubling, halving, or retaining the current step bound to get δ_+. The actual conditions are somewhat arbitrary; what is important is that if our current quadratic model is predicting the function well, we increase the trust region, but if it is predicting poorly, we decrease the trust region. If the quadratic model has predicted the actual function reduction sufficiently well—i.e., $\Delta f \leq 0.75\,\Delta f_{\text{pred}}$—then we take $\delta_+ = 2\delta_c$. If the model has greatly overestimated the decrease in $f(x)$—i.e., $\Delta f > 0.1\,\Delta f_{\text{pred}}$—then we take $\delta_+ = \delta_c/2$. Otherwise $\delta_+ = \delta_c$.

EXAMPLE 6.4.5 Let $f(x)$, x_c, H_c, δ_c be given by Example 6.4.3, and suppose we have just taken the step determined in that example,

$$x_+ = x_c + s_c = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 0.334 \\ 0.335 \end{pmatrix} = \begin{pmatrix} 0.666 \\ 0.665 \end{pmatrix}.$$

Recall that

$$\nabla f(x_c) = (6, 2)^T, \qquad H_c = \nabla^2 f(x_c) = \begin{bmatrix} 14 & 0 \\ 0 & 2 \end{bmatrix}.$$

We want to decide whether x_+ is a satisfactory point, and update the trust region. First, we calculate

$$f(x_+) = 1.083,$$

$$f(x_c) + \alpha\, \nabla f(x_c)^T (x_+ - x_c) = 3 - 10^{-4}(2.673) = 2.9997.$$

Therefore, x_+ is acceptable. Next, we decide whether to try a larger step at the current iteration. We calculate

$$\Delta f = f(x_+) - f(x_c) = -1.917$$

and

$$\Delta f_{\text{pred}} = m_c(x_+) - f(x_c) = \nabla f(x_c)^T s_c + \tfrac{1}{2} s_c^T H_c s_c$$
$$= -2.673 + 0.892 = -1.781.$$

Since $|\Delta f - \Delta f_{\text{pred}}| / |\Delta f| = 0.071 < 0.1$, we double the trust region and go back to the locally constrained optimal step, Algorithm A6.4.1. The reader can confirm that with the new trust radius $\delta_c = 1$, Algorithm A6.4.1 will select the Newton step. It is an exercise to complete the updating of the trust region for this global iteration.

The algorithm for updating the trust region is Algorithm A6.4.5 in the appendix. It has several additional features.

1. It uses a minimum and maximum step length, discussed in Section 7.2. The trust radius is never allowed outside these bounds. The maximum step length is supplied by the user. The minimum step length is the quantity used to test for convergence in Algorithm 7.2.1. If x_+ is unsatisfactory, but the current trust region is already less than *minstep*, the global step is halted, because convergence would necessarily be detected at the end of the current global iteration. This situation may indicate convergence to a nonsolution, so a warning message should be printed.
2. When the approximate solution from Algorithm A6.4.2 or A6.4.4 to the constrained model problem is a Newton step that is shorter than the current trust radius, these algorithms immediately reduce the size of the trust region to the length of the Newton step. It is then still adjusted by Algorithm A6.4.5. This is an additional mechanism for regulating the trust region.
3. The algorithm is implemented using a diagonal scaling of the available space, discussed in Section 7.1.

Finally, we discuss how the initial estimate of the trust region radius, or step bound is obtained. Sometimes the user can supply a reasonable estimate based on his knowledge of the problem. If not, Powell (1970a) suggests using the length of the Cauchy step (see Subsection 6.4.2) as the initial trust region radius. Other strategies are possible. The updating strategy of Algorithm A6.4.5 does enable a trust region algorithm to recover in practice from a bad starting value of δ, but there is usually some cost in additional iterations. Therefore the initial trust region is reasonably important.

6.5 GLOBAL METHODS FOR SYSTEMS OF NONLINEAR EQUATIONS

We return now to the nonlinear equations problem:

$$\begin{aligned} &\text{given} \quad F\colon \mathbb{R}^n \longrightarrow \mathbb{R}^n, \\ &\text{find} \quad x_* \in \mathbb{R}^n \quad \text{such that} \quad F(x_*) = 0, \end{aligned} \tag{6.5.1}$$

In this section we show how Newton's method for (6.5.1) can be combined with global methods for unconstrained optimization to produce global methods for (6.5.1).

The Newton step for (6.5.1) is

$$x_+ = x_c - J(x_c)^{-1}F(x_c), \tag{6.5.2}$$

where $J(x_c)$ is the Jacobian matrix of F at x_c. From Section 5.2 we know that (6.5.2) is locally q-quadratically convergent to x_*, but not necessarily globally convergent. Now assume x_c is not close to any solution x_* of (6.5.1). How would one decide then whether to accept x_+ as the next iterate? A reasonable answer is that $\|F(x_+)\|$ should be less than $\|F(x_c)\|$ for some norm $\|\cdot\|$, a convenient choice being the l_2 norm $\|F(x)\|_2^2 = F(x)^T F(x)$.

Requiring that our step result in a decrease of $\|F(x)\|_2$ is the same thing we would require if we were trying to find a minimum of the function $\|F(x)\|_2$. Thus, we have in effect turned our attention to the corresponding minimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) = \tfrac{1}{2}F(x)^T F(x), \tag{6.5.3}$$

where the "$\frac{1}{2}$" is added for later algebraic convenience. Note that every solution to (6.5.1) is a solution to (6.5.3), but there may be local minimizers of (6.5.3) that are not solutions to (6.5.1) (see Figure 6.5.1). Therefore, although we could try to solve (6.5.1) simply by using a minimization routine on (6.5.3), it is better to use the structure of the orginal problem wherever possible, in particular to compute the Newton step (6.5.2). However, our global strategy for (6.5.1) will be based on a global strategy for the related minimization problem (6.5.3).

An important question to ask is, "What is a descent direction for prob-

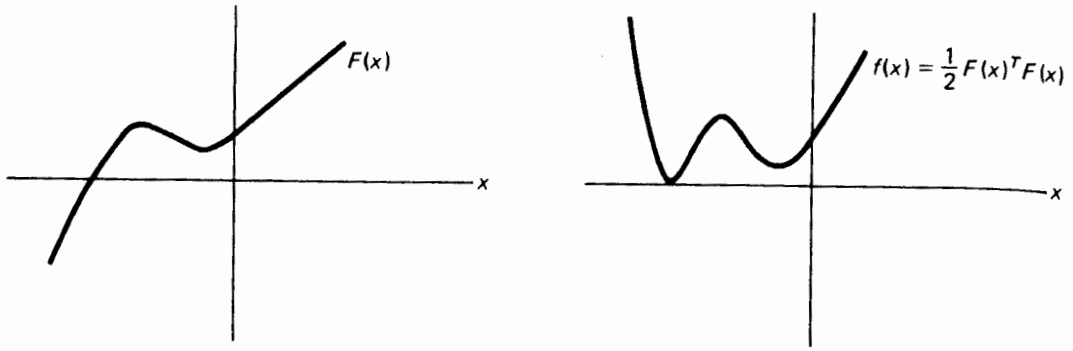


Figure 6.5.1 The nonlinear equations and corresponding minimization problem, in one dimension

lem (6.5.3)?" It is any direction p for which $\nabla f(x_c)^T p < 0$, where

$$\nabla f(x_c) = \frac{d}{dx} \sum_{i=1}^{n} \tfrac{1}{2}(f_i(x_c))^2 = \sum_{i=1}^{n} \nabla f_i(x_c) \cdot f(x_c) = J(x_c)^T F(x_c).$$

Therefore, the steepest-descent direction for (6.5.3) is along $-J(x_c)^T F(x_c)$. Furthermore, the Newton direction along $s^N = -J(x_c)^{-1} F(x_c)$ is a descent direction, since

$$\nabla f(x_c)^T s^N = -F(x_c)^T J(x_c) J(x_c)^{-1} F(x_c) = -F(x_c)^T F(x_c) < 0$$

as long as $F(x_c) \neq 0$. This may seem surprising, but it is geometrically reasonable. Since the Newton step yields a root of

$$M_c(x_c + s) = F(x_c) + J(x_c)s,$$

it also goes to a minimum of the quadratic function

$$\begin{aligned} \hat{m}_c(x_c + s) &\triangleq \tfrac{1}{2} M_c(x_c + s)^T M_c(x_c + s) \\ &= \tfrac{1}{2} F(x_c)^T F(x_c) + (J(x_c)^T F(x_c))^T s + \tfrac{1}{2} s^T (J(x_c)^T J(x_c)) s, \end{aligned} \tag{6.5.4}$$

because $\hat{m}_c(x_c + s) \geq 0$ for all s and $\hat{m}_c(x_c + s^N) = 0$. Therefore, s^N is a descent direction for $\hat{m}_c$, and since the gradients at x_c of $\hat{m}_c$ and f are the same, it is also a descent direction for f.

The above development motivates how we will create global methods for (6.5.1). They will be based on applying our algorithms of Section 6.3 or 6.4 to the quadratic model $\hat{m}_c(x)$ in (6.5.4). Since $\nabla^2 \hat{m}(x_c) = J(x_c)^T J(x_c)$, this model is positive definite as long as $J(x_c)$ is nonsingular, which is consistent with the fact that $x_c + s^N$ is the unique root of $M_c(x)$ and thus the unique minimizer of $\hat{m}_c(x)$ in this case. Thus, the model $\hat{m}_c(x)$ has the attractive properties that its minimizer is the Newton point for the original problem, and that all its descent directions are descent directions for $f(x)$ because $\nabla \hat{m}_c(x_c) = \nabla f(x_c)$. Therefore methods based on this model, by going downhill and trying to

minimize $\hat{m}_c(x)$, will combine Newton's method for nonlinear equations with global methods for an associated minimization problem. Note that $\hat{m}_c(x)$ is not quite the same as the quadratic model of $f(x) = \frac{1}{2}F(x)^T F(x)$ around x_c,

$$m_c(x_c + s) = f(x_c) + \nabla f(x_c)^T s + \tfrac{1}{2} s^T \nabla^2 f(x_c) s,$$

because $\nabla^2 f(x_c) \neq J(x_c)^T J(x_c)$ (see Exercise 18 of Chapter 5).

The application of the global methods of Sections 6.3 and 6.4 to the nonlinear equations problem is now straightforward. As long as $J(x_c)$ is sufficiently well conditioned, then $J(x_c)^T J(x_c)$ is safely positive definite, and the algorithms apply without change if we define the objective function by $\frac{1}{2}\|F(x)\|_2^2$, the Newton direction by $-J(x)^{-1}F(x)$, and the positive definite quadratic model by (6.5.4). This means that in a line-search algorithm, we search in the Newton direction, looking for a sufficient decrease in $\|F(x)\|_2$. In trust region algorithms, we approximately minimize $\hat{m}_c(x_c + s)$ subject to $\|s\|_2 \leq \delta_c$. If $\delta_c \geq \|J(x_c)^{-1}F(x_c)\|_2$, then the step attempted is the Newton step; otherwise, for the locally constrained optimal step, it is

$$s = -(J(x_c)^T J(x_c) + \mu_c I)^{-1} J(x_c)^T F(x_c) \tag{6.5.5}$$

for μ_c such that $\|s\|_2 \cong \delta_c$. Using either global strategy, we expect to be taking Newton steps for $F(x) = 0$ eventually. Example 6.5.1 shows how our global strategy using a line search would work on the example of Section 5.4, started here from a different point, one at which the Newton step is unsatisfactory.

EXAMPLE 6.5.1 Let $F: \mathbb{R}^2 \longrightarrow \mathbb{R}^2$,

$$F(x) = \begin{bmatrix} x_1^2 + x_2^2 - 2 \\ e^{x_1 - 1} + x_2^3 - 2 \end{bmatrix},$$

which has the root $x_+ = (1, 1)^T$, and let $x_0 = (2, 0.5)^T$. Define $f(x) = \frac{1}{2}F(x)^T F(x)$. Then

$$J(x_0) = \begin{bmatrix} 4 & 1 \\ e & 0.75 \end{bmatrix}, \qquad F(x_0) \cong \begin{bmatrix} 2.25 \\ 0.843 \end{bmatrix},$$

and

$$s_0^N = -J(x_0)^{-1}F(x_0) \cong \begin{bmatrix} -3.00 \\ 9.74 \end{bmatrix}.$$

Our line-search algorithm A6.3.1 will calculate $x_+ = x_0 + \lambda_0 s_0^N$ starting with $\lambda_0 = 1$, decreasing λ_0 if necessary until $f(x_+) < f(x_0) + 10^{-4}\lambda_0 \nabla f(x_0)^T s_0^N$. For $\lambda_0 = 1$,

$$x_+ = x_0 + s_0^N \cong \begin{bmatrix} -1.00 \\ 10.24 \end{bmatrix}, \qquad F(x_+) \cong \begin{bmatrix} 104 \\ 1071 \end{bmatrix},$$

so that the Newton step clearly is unsatisfactory. Therefore we reduce λ_0 by a

quadratic backtrack, calculating

$$\lambda_1 = \frac{-\nabla f(x_0)^T s_0^N}{2[f(x_+) - f(x_0) - \nabla f(x)^T s_0^N]}. \tag{6.5.6}$$

In this case, $f(x_+) \cong 5.79 \times 10^5$, $f(x_0) \cong 2.89$, $\nabla f(x)^T s_0^N = -F(x_0)^T F(x_0) \cong -5.77$, so that (6.5.6) gives $\lambda_1 \cong 4.99 \times 10^{-6}$. Since $\lambda_1 < 0.1$, Algorithm A6.3.1 sets $\lambda_1 = 0.1$,

$$x_+ = x_0 + 0.1 s_0^N \cong \begin{bmatrix} 1.70 \\ 1.47 \end{bmatrix}, \qquad F(x_+) \cong \begin{bmatrix} 3.06 \\ 3.21 \end{bmatrix}.$$

This is still unsatisfactory, and so Algorithm A6.3.2 does a cubic backtrack. The reader can verify that a backtrack yields $\lambda_2 \cong 0.0659$. Since $\lambda_2 > \frac{1}{2}\lambda_1$, the algorithm sets $\lambda_2 = \frac{1}{2}\lambda_1 = 0.05$,

$$x_+ = x_0 + 0.05 s_0^N \cong \begin{bmatrix} 1.85 \\ 0.987 \end{bmatrix}, \qquad F(x_+) \cong \begin{bmatrix} 2.40 \\ 1.30 \end{bmatrix}.$$

This point is still unsatisfactory, since $f(x_+) \cong 3.71 > f(x_0)$, so the algorithm does another cubic backtrack. It yields $\lambda_3 \cong 0.0116$, which is used since it is in the interval $[\lambda_2/10, \lambda_2/2] = [0.005, 0.025]$. Now

$$x_+ = x_0 + 0.0116 s_0^N \cong \begin{bmatrix} 1.965 \\ 0.613 \end{bmatrix}, \qquad F(x_+) \cong \begin{bmatrix} 2.238 \\ 0.856 \end{bmatrix}.$$

This point is satisfactory, since

$$f(x_+) \cong 2.87 < 2.89 \cong f(x_0) + 10^{-4}(0.0116)\, \nabla f(x_0)^T s_0^N,$$

so we set $x_1 = x_+$ and proceed to the next iteration.

It is interesting to follow this problem further. At the next iteration,

$$s_1^N = -J(x_1)^{-1} F(x_1) \cong \begin{bmatrix} -1.22 \\ 2.07 \end{bmatrix},$$

$$x_2^N = x_1 + s_1^N \cong \begin{bmatrix} 0.750 \\ 2.68 \end{bmatrix}, \qquad F(x_2^N) \cong \begin{bmatrix} 5.77 \\ 18.13 \end{bmatrix},$$

which is again unsatisfactory. However, the first backtrack step is successful: Algorithm A6.31 computes $\lambda_1 = 0.0156$, and, since this is less than 0.1, it sets $\lambda_1 = 0.1$,

$$x_+ = x_1 + 0.1 s_1^N \cong \begin{bmatrix} 1.84 \\ 0.820 \end{bmatrix}, \qquad F(x_+) \cong \begin{bmatrix} 2.07 \\ 0.876 \end{bmatrix},$$

and the step is accepted, since

$$f(x_+) \cong 2.53 < 2.87 \cong f(x_1) + 10^{-4}(0.1)\, \nabla f(x_1)^T s_1^N.$$

At the next iteration,

$$s_2^N = -J(x_2)^{-1}F(x_2) \cong \begin{bmatrix} -0.0756 \\ 0.0437 \end{bmatrix},$$

$$x_3^N = x_2 + s_2^N \cong \begin{bmatrix} 1.088 \\ 1.257 \end{bmatrix}, \qquad F(x_3^N) \cong \begin{bmatrix} 0.762 \\ 1.077 \end{bmatrix},$$

so the Newton step is very good. From here on, Newton's method converges q-quadratically to $x_* = (1, 1)^T$.

The only complication to this global strategy arises when J is nearly singular at the current point x_c. In this case we cannot accurately calculate the Newton direction $s^N = -J(x_c)^{-1}F(x_c)$, and the model Hessian $J(x_c)^T J(x_c)$ is nearly singular. To detect this situation, we perform the QR factorization of $J(x_c)$, and if R is nonsingular, estimate its condition number by Algorithm A3.3.1. If R is singular or its estimated condition number is greater than macheps$^{-1/2}$, we perturb the quadratic model to

$$\hat{m}_c(x_c + s) = \tfrac{1}{2}F(x_c)^T F(x_c) + (J(x_c)^T F(x_c))^T s + \tfrac{1}{2}s^T H_c\, s,$$

where

$$H_c = J(x_c)^T J(x_c) + (n \cdot \text{macheps})^{1/2} \| J(x_c)^T J(x_c) \|_1 \cdot I.$$

(The condition number of H_c is about macheps$^{-1/2}$; see Exercise 23 for a more precise justification.) From Section 6.4 we know that the Newton step to the minimizer of this model, $s^N = -H_c^{-1} J(x_c)^T F(x_c)$, solves

$$\min_{s \in \mathbb{R}^n} \hat{m}_c(x_c + s) = \| J(x_c)s + F(x_c) \|_2^2$$

$$\text{subject to} \qquad \| s \|_2 \leq \delta$$

for some $\delta > 0$. It is also a descent direction for $f(x) = \frac{1}{2}\| F(x) \|_2^2$. Therefore, we prefer this modified Newton step to some $s = -\hat{J}(x_c)^{-1}F(x_c)$ that would result from a perturbation $\hat{J}(x_c)$ of the QR factorization of $J(x_c)$. See Exercise 24 for further discussion of this step.

Driver D6.1.3 shows how we use the global algorithms for unconstrained minimization in solving systems of nonlinear equations. First at each iteration, Algorithm A6.5.1 calculates the $Q_c R_c$ factorization of $J(x_c)$ and estimates the condition number of R_c. If the condition number estimate is less than macheps$^{-1/2}$, it calculates the Newton step $-J(x_c)^{-1}F(x_c)$, otherwise it calculates the perturbed Newton step described above. Then the driver calls either the line-search, dogleg, or locally constrained optimal algorithm as selected by the user. All these global algorithms require $\nabla f(x_c) = J(x_c)^T F(x_c)$, which is also calculated by Algorithm D6.1.3. In addition, the trust region algorithms re-

quire the Cholesky factorization $L_c L_c^T$ of $H_c = J(x_c)^T J(x_c)$ (except for the case when H_c is perturbed as described above). Since we have $J(x_c) = Q_c R_c$, it is immediate that $L_c = R_c^T$. Finally, all our algorithms for nonlinear equations are implemented using a diagonal scaling matrix D_F on $F(x)$, discussed in Section 7.2. It causes

$$f(x) = \tfrac{1}{2}\| D_F F(x)\|_2^2, \qquad \nabla f(x) = J(x)^T D_F^2 F(x), \quad \text{and} \quad H = J(x)^T D_F^2 J(x).$$

There is one significant case when these global algorithms for nonlinear equations can fail. It is when a local minimizer of $f(x) = \frac{1}{2}\| F(x)\|_2^2$ is not a root of $F(x)$ (see Figure 6.5.1). A global minimization routine, started close to such a point, may converge to it. There is not much one can do in such a case, except report what has happened and advise the user to try to restart nearer to a root of $F(x)$ if possible. The stopping routine for nonlinear equations (Algorithm A7.2.3) detects this situation and produces such a message. Research is currently underway on methods that might be able to restart themselves from such local minimizers. See Allgower and Georg (1980) and Zirilli (1982).

6.6 EXERCISES

1. Let $f(x) = 3x_1^2 + 2x_1x_2 + x_2^2$, $x_0 = (1, 1)^T$. What is the steepest-descent direction for f from x_0? Is $(1, -1)^T$ a descent direction?
2. Show that for a positive definite quadratic $f(x_k + s) = g^T s + \frac{1}{2}s^T \nabla^2 f(x_k)s$, the steepest-descent step from x_k is given by equation (6.2.5).
3. Given a $\hat{c} \in (0, 1)$, generalize Example 6.2.2 to an example where the sequence of points generated by the steepest-descent algorithm from a specific x_0 obeys $\| x_{k+1} - x_*\| = \hat{c}\| x_k - x_*\|$, $k = 0, 1, \ldots$.
4. Let $f(x) = x^2$. Show that the infinite sequence of points from Figure 6.3.2, $x_i = (-1)^i(1 + 2^{-i})$, $i = 0, 1, \ldots$, is not permitted by (6.3.3) for any $\alpha > 0$.
5. Let $f(x) = x^2$. Show that the infinite sequence of points $x_i = 1 + 2^{-i}$, $i = 0, 1, \ldots$, is permitted by (6.3.3) for any $\alpha \le \frac{1}{2}$ but is prohibited by (6.3.4) for any $\beta > 0$.
6. Show that if $f(x)$ is a positive definite quadratic, the Newton step from any $x_k \in \mathbb{R}^n$ satisfies condition (6.3.3) for $\alpha \le \frac{1}{2}$ and (6.3.4) for any $\beta > 0$.
7. Prove that if (6.3.4) is replaced by

$$f(x_{k+1}) \ge f(x_k) + \beta \nabla f(x_k)^T(x_{k+1} - x_k), \qquad \beta \in (\alpha, 1),$$

then Theorems 6.3.2 and 6.3.3 remain true; and that if in addition $\beta > \frac{1}{2}$, then Theorem 6.3.4 also remains true. This is Goldstein's original condition.

8. Let $f(x) = x_1^4 + x_2^2$, $x_c = (1, 1)^T$, and the search direction p_c be determined by the Newton step. What will x_+ be, using:
   (a) The Newton step?
   (b) Line-search Algorithm A6.3.1?
   (c) A "perfect line-search" algorithm that sets x_+ to the minimizer of $f(x)$ in the direction p_c from x_c? [Answer (c) approximately.]

9. Let $f(x) = \frac{1}{2}x_1^2 + x_2^2$, $x_c = (1, 1)^T$. What is the exact solution to (6.4.1) if $\delta = 2$? if $\delta = \frac{5}{6}$? [*Hint*: Try $\mu = 1$ for the second part.]

10. Prove that the locally constrained optimal curve $s(\mu)$ in Figure 6.4.2 is not necessarily planar by constructing a simple three-dimensional example in which x_c and some three points on the curve are not coplanar. [$f(x) = x_1^2 + 2x_2^2 + 3x_3^2$ will do.]

11. Let $H \in \mathbb{R}^{n\times n}$ be symmetric and positive definite, and let $v_1, \ldots, v_n$ be an orthonormal basis of eigenvectors for H with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$. Show that for

$$g = \sum_{j=1}^{n} \alpha_j v_j \quad \text{and} \quad \mu \geq 0, \qquad (H + \mu I)^{-1} g = \sum_{j=1}^{n} \left(\frac{\alpha_j}{\lambda_j + \mu}\right) v_j.$$

How does this relate to the model $m_c(\mu)$ for $\|H + \mu I^{-1} g\| - \delta$ used in the locally constrained optimal algorithm?

12. Let H, g be given by Exercise 11, and define $s(\mu) = (H + \mu I)^{-1} g$ for $\mu \geq 0$ and $\eta(\mu) = \|s(\mu)\|_2$. Using the techniques of Exercise 11, show that

$$\frac{d}{d\mu}\eta(\mu) = -\frac{s(\mu)^T (H + \mu I)^{-1} s(\mu)}{\|s(\mu)\|_2}.$$

[You can also show that $(d^2/d\mu^2)\eta(\mu) > 0$ for all $\mu > 0$ using these techniques, but the proof is tedious.]

13. Let $f(x) = \frac{1}{2}x_1^2 + x_2^2$, $x_0 = (1, 1)^T$, $g = \nabla f(x_0)$, $H = \nabla^2 f(x_0)$. Calculate the Cauchy point of f from x_0 (see Subsection 6.4.2), and the point $\hat{N} = x_0 - (0.8\gamma + 0.2)H^{-1}g$ as used in our double dogleg algorithm. Then graph the double dogleg curve and show graphically the values of x_1 if $\delta = 1$; if $\delta = \frac{5}{4}$.

14. Let $H \in \mathbb{R}^{n\times n}$ be symmetric and positive definite, $g \in \mathbb{R}^n$. Prove that $(g^T g)^2 \leq (g^T H g)(g^T H^{-1} g)$. [*Hint:* Let $u = H^{1/2} g$, $v = H^{-1/2} g$, and apply the Cauchy-Schwarz inequality.]

15. Complete Example 6.4.5.

16. Let $F(x) = (x_1, 2x_2)^T$, $x_0 = (1, 1)^T$. Using the techniques of Section 6.5, what is the "steepest-descent" step from x_0 for $F = 0$? If all your steps were in steepest-descent directions, what rate of convergence to the root of F would you expect?

17. Let $F: \mathbb{R}^n \longrightarrow \mathbb{R}^n$ be continuously differentiable and $\hat{x} \in \mathbb{R}^n$. Suppose $\hat{x}$ is a local minimizer of $f(x) \triangleq \frac{1}{2}F(x)^T F(x)$ but $F(\hat{x}) \neq 0$. Is $J(\hat{x})$ singular? If $J(\hat{x})$ is singular, must $\hat{x}$ be a local minimizer of $f(x)$?

18. An alternative quadratic model for $F = 0$ to the one used in the global algorithms of Section 6.5 is the first three terms of the Taylor series of $\frac{1}{2}\|F(x_c + s)\|_2^2$. Show that this model is

$$m_c(x_c + s) = \tfrac{1}{2}F(x_c)^T F(x_c) + [J(x_c)^T F(x_c)]^T s + \tfrac{1}{2}s^T [J(x_c)^T J(x_c) + \sum_{i=1}^{n} f_i(x_c)\, \nabla^2 f_i(x_c)]s.$$

How does this compare with the model used in Section 6.5? Is the Newton step for minimizing $m_c(x)$ the same as the Newton step for $F(x) = 0$? Comment on the attractiveness of this model versus the one used in Section 6.5. (An analogous

situation with different conclusions occurs in the solution of nonlinear least-squares problems—see Chapter 10.)

19. Why is the Newton step from x_0 in Example 6.5.1 bad? [*Hint*: What would happen if x_0 were changed to $(2, e/6)^T \cong (2, 0.45)^T$?]

20. What step would be taken from x_0 in Example 6.5.1 using the double dogleg strategy (Algorithm A6.4.4) with $\delta_0 = \frac{1}{2}$?

21. Find out what the conjugate gradient algorithm is for minimizing convex quadratic functions, and show that the dogleg with $\hat{N} = x_+^N$ is the conjugate gradient method applied to $m_c(x)$ on the subspace spanned by the steepest-descent and Newton directions.

22. One of the disadvantages often cited for the dogleg algorithm is that its steps are restricted to be in the two-dimensional subspace spanned by the steepest-descent and Newton directions at each step. Suggest ways to alleviate this criticism, based on Exercise 21. See Steihaug (1981).

23. Let $J \in \mathbb{R}^{n \times n}$ be singular. Show that

$$\frac{1}{n(\text{macheps})^{1/2}} \leq \kappa_2(J^TJ + (n \cdot \text{macheps})^{1/2} \| J^TJ \|_1 I) - 1 \leq \frac{1}{(\text{macheps})^{1/2}}.$$

[*Hint*: Use (3.1.13) and Theorem 3.5.7.] Generalize this inequality to the case when $\kappa_2(J) \geq \text{macheps}^{-1/2}$.

24. Let $F \in \mathbb{R}^n$ be nonzero, $J \in \mathbb{R}^{n \times n}$ singular. Another step that has been suggested for the nonlinear equations problem is the s that solves

$$\min \{\|s\|_2 : s \text{ solves } \min_{s \in \mathbb{R}^n} \|F + Js\|_2\}. \tag{6.6.1}$$

It was shown in Section 3.6 that the solution to (6.6.1) is $s = -J^+F$, where J^+ is the pseudoinverse of J. Show that this step is similar to $\hat{s} = -(J^TJ + \alpha I)^{-1}J^TF$, where $\alpha > 0$ is small, by proving:

(a) $\lim_{\alpha \to 0+} (J^TJ + \alpha I)^{-1}J^T = J^+$.

(b) for any $\alpha > 0$ and $v \in \mathbb{R}^n$, both $(J^TJ + \alpha I)^{-1}J^Tv$ and J^+v are perpendicular to all vectors w in the null space of J.

25. The global convergence of the line search and trust region algorithms of Subsections 6.3.2, 6.4.1, and 6.4.2 does not follow directly from the theory in Subsection 6.3.1, because none of the algorithms implement condition (6.3.4). However, the bounds in the algorithms on the amount of each adjustment in the line search parameter λ or the trust region δ_c have the same theoretical effect. For a proof that all of our algorithms are globally convergent, see Shultz, Schnabel, and Byrd (1982).