1-2 - Fundamentals

Convexity

The term "convex" can be applied both to sets and to functions. A set $S \in \mathbb{R}^n$ is a *convex set* if the straight line segment connecting any two points in S lies entirely inside S. Formally, for any two points $x \in S$ and $y \in S$, we have $\alpha x + (1-\alpha)y \in S$ for all $\alpha \in [0, 1]$. The function f is a *convex function* if its domain S is a convex set and if for any two points x and y in S, the following property is satisfied:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y), \qquad \text{for all } \alpha \in [0, 1]. \tag{1.4}$$

Theorem 2.1 (Taylor's Theorem).
Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and that $p \in \mathbb{R}^n$. Then we have that

$$f(x + p) = f(x) + \nabla f(x + tp)^T p, \tag{2.4}$$

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

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

and that

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

for some $t \in (0, 1)$.

Matrixes

A square matrix A is *positive definite* if there is a positive scalar α such that

$$x^T A x \geq \alpha x^T x, \qquad \text{for all } x \in \mathbb{R}^n. \tag{A.1}$$

It is *positive semidefinite* if

$$x^T A x \geq 0, \qquad \text{for all } x \in \mathbb{R}^n.$$

We can recognize that a symmetric matrix is positive definite by computing its eigenvalues and verifying that they are all positive, or by performing a Cholesky factorization. Both techniques are discussed further in later sections.

A square $n \times n$ matrix A is *nonsingular* if for any vector $b \in \mathbb{R}^n$, there exists $x \in \mathbb{R}^n$ such that $Ax = b$. For nonsingular matrices A, there exists a unique $n \times n$ matrix B such that $AB = BA = I$. We denote B by A^{-1} and call it the *inverse* of A. It is not hard to show that the inverse of A^T is the transpose of A^{-1}.

A square matrix Q is *orthogonal* if it has the property that $QQ^T = Q^T Q = I$. In other words, the inverse of an orthogonal matrix is its transpose.

Cholesky factorization

When $A \in \mathbb{R}^{n \times n}$ is symmetric positive definite, it is possible to compute a similar but more specialized factorization at about half the cost—about $n^3/3$ operations. This factorization, known as the Cholesky factorization, produces a matrix L such that

$$A = LL^T. \tag{A.23}$$

(If we require L to have positive diagonal elements, it is uniquely defined by this formula.)

A point x^* is a *local minimizer* if there is a neighborhood $\mathcal{N}$ of x^* such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{N}$.

Theorem 2.2 (First-Order Necessary Conditions).
If x^* is a local minimizer and f is continuously differentiable in an open neighborhood of x^*, then $\nabla f(x^*) = 0$.

We call x^* a *stationary point* if $\nabla f(x^*) = 0$. According to Theorem 2.2, any local minimizer must be a stationary point.

For the next result we recall that a matrix B is positive definite if $p^T Bp > 0$ for all $p \neq 0$, and positive semidefinite if $p^T Bp \geq 0$ for all p (see the Appendix).

Theorem 2.3 (Second-Order Necessary Conditions).
If x^* is a local minimizer of f and $\nabla^2 f$ exists and is continuous in an open neighborhood of x^*, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.

Theorem 2.4 (Second-Order Sufficient Conditions).
Suppose that $\nabla^2 f$ is continuous in an open neighborhood of x^* and that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite. Then x^* is a strict local minimizer of f.

Theorem 2.5.
When f is convex, any local minimizer x^* is a global minimizer of f. If in addition f is differentiable, then any stationary point x^* is a global minimizer of f.

RATES OF CONVERGENCE

One of the key measures of performance of an algorithm is its rate of convergence. Here, we define the terminology associated with different types of convergence.

Let $\{x_k\}$ be a sequence in $\mathbb{R}^n$ that converges to x^*. We say that the convergence is *Q-linear* if there is a constant $r \in (0, 1)$ such that

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \leq r, \qquad \text{for all } k \text{ sufficiently large.} \tag{A.34}$$

This means that the distance to the solution x^* decreases at each iteration by at least a constant factor bounded away from 1. For example, the sequence $1 + (0.5)^k$ converges Q-linearly to 1, with rate $r = 0.5$. The prefix "Q" stands for "quotient," because this type of convergence is defined in terms of the quotient of successive errors.

The convergence is said to be *Q-superlinear* if

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

where M is a positive constant, not necessarily less than 1. An example is the sequence $1 + (0.5)^{2^k}$.

The speed of convergence depends on r and (more weakly) on M, whose values depend not only on the algorithm but also on the properties of the particular problem. Regardless of these values, however, a quadratically convergent sequence will always eventually converge faster than a linearly convergent sequence.

Obviously, any sequence that converges Q-quadratically also converges Q-superlinearly, and any sequence that converges Q-superlinearly also converges Q-linearly. We can also define higher rates of convergence (cubic, quartic, and so on), but these are less interesting in practical terms. In general, we say that the Q-order of convergence is p (with $p > 1$) if there is a positive constant M such that

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

Lipschitz

Considering again the general case of $f : \mathcal{D} \to \mathbb{R}^m$ where $\mathcal{D} \subset \mathbb{R}^n$ for general m and n. The function f is said to be *Lipschitz continuous* on some set $\mathcal{N} \subset \mathcal{D}$ if there is a constant $L > 0$ such that

$$\|f(x_1) - f(x_0)\| \leq L\|x_1 - x_0\|, \qquad \text{for all } x_0, x_1 \in \mathcal{N}. \tag{A.42}$$

(L is called the *Lipschitz constant.*) The function f is *locally Lipschitz continuous* at a point $\bar{x} \in \text{int}\mathcal{D}$ if there is some neighborhood $\mathcal{N}$ of $\bar{x}$ with $\mathcal{N} \subset \mathcal{D}$ such that the property (A.42) holds for some $L > 0$.

If g and h are two functions mapping $\mathcal{D} \subset \mathbb{R}^n$ to $\mathbb{R}^m$, Lipschitz continuous on a set $\mathcal{N} \subset \mathcal{D}$, their sum $g + h$ is also Lipschitz continuous, with Lipschitz constant equal to the sum of the Lipschitz constants for g and h individually. If g and h are two functions mapping $\mathcal{D} \subset \mathbb{R}^n$ to $\mathbb{R}$, the product gh is Lipschitz continuous on a set $\mathcal{N} \subset \mathcal{D}$ if both g and h are Lipschitz continuous on $\mathcal{N}$ and both are bounded on $\mathcal{N}$ (that is, there is $M > 0$ such that $|g(x)| \leq M$ and $|h(x)| \leq M$ for all $x \in \mathcal{N}$).

MEAN VALUE THEOREM

We now recall the mean value theorem for univariate functions. Given a continuously differentiable function $\phi : \mathbb{R} \to \mathbb{R}$ and two real numbers α_0 and α_1 that satisfy $\alpha_1 > \alpha_0$, we have that

$$\phi(\alpha_1) = \phi(\alpha_0) + \phi'(\xi)(\alpha_1 - \alpha_0) \tag{A.54}$$

for some $\xi \in (\alpha_0, \alpha_1)$. An extension of this result to a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$ is that for any vector p we have

$$f(x + p) = f(x) + \nabla f(x + \alpha p)^T p, \tag{A.55}$$

for some $\alpha \in (0, 1)$. (This result can be proved by defining $\phi(\alpha) = f(x + \alpha p)$, $\alpha_0 = 0$, and $\alpha_1 = 1$ and applying the chain rule, as above.)

3 - Line-Search Methods

Each iteration of a line search method computes a search direction p_k and then decides how far to move along that direction. The iteration is given by

$$x_{k+1} = x_k + \alpha_k p_k, \tag{3.1}$$

$$p_k = -B_k^{-1} \nabla f_k, \tag{3.2}$$

where B_k is a symmetric and nonsingular matrix. In the steepest descent method, B_k is simply the identity matrix I, while in Newton's method, B_k is the exact Hessian $\nabla^2 f(x_k)$. In quasi-Newton methods, B_k is an approximation to the Hessian that is updated at every iteration by means of a low-rank formula. When p_k is defined by (3.2) and B_k is positive definite, we have

$$p_k^T \nabla f_k = -\nabla f_k^T B_k^{-1} \nabla f_k < 0,$$

3.1 STEP LENGTH

In computing the step length α_k, we face a tradeoff. We would like to choose α_k to give a substantial reduction of f, but at the same time we do not want to spend too much time making the choice. The ideal choice would be the global minimizer of the univariate function $\phi(\cdot)$ defined by

$$\phi(\alpha) = f(x_k + \alpha p_k), \quad \alpha > 0, \tag{3.3}$$

Newton's step

$$p_k^N = -\nabla^2 f_k^{-1} \nabla f_k. \tag{3.30}$$

THE WOLFE CONDITIONS

A popular inexact line search condition stipulates that α_k should first of all give *sufficient decrease* in the objective function f, as measured by the following inequality:

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

for some constant $c_1 \in (0, 1)$. In other words, the reduction in f should be proportional to both the step length α_k and the directional derivative $\nabla f_k^T p_k$. Inequality (3.4) is sometimes called the *Armijo condition.*

The sufficient decrease condition is not enough by itself to ensure that the algorithm makes reasonable progress because, as we see from Figure 3.3, it is satisfied for all sufficiently small values of α. To rule out unacceptably short steps we introduce a second requirement, called the *curvature condition*, which requires α_k to satisfy

$$\nabla f(x_k + \alpha_k p_k)^T p_k \ge c_2 \nabla f_k^T p_k, \tag{3.5}$$

with $0 < c_1 < c_2 < 1$.

A step length may satisfy the Wolfe conditions without being particularly close to a minimizer of ϕ, as we show in Figure 3.5. We can, however, modify the curvature condition to force α_k to lie in at least a broad neighborhood of a local minimizer or stationary point of ϕ. The *strong Wolfe conditions* require α_k to satisfy

$$f(x_k + \alpha_k p_k) \le f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, \tag{3.7a}$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \le c_2 |\nabla f_k^T p_k|, \tag{3.7b}$$

with $0 < c_1 < c_2 < 1$. The only difference with the Wolfe conditions is that we no longer allow the derivative $\phi'(\alpha_k)$ to be too positive. Hence, we exclude points that are far from stationary points of ϕ.

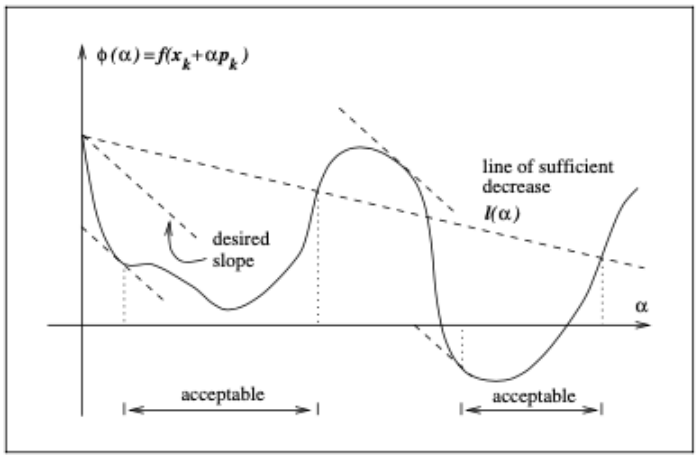

Figure 3.5 Step lengths satisfying the Wolfe conditions.

THE GOLDSTEIN CONDITIONS

Like the Wolfe conditions, the *Goldstein conditions* ensure that the step length α achieves sufficient decrease but is not too short. The Goldstein conditions can also be stated as a pair of inequalities, in the following way:

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

with $0 < c < 1/2$. The second inequality is the sufficient decrease condition (3.4), whereas the first inequality is introduced to control the step length from below; see Figure 3.6

A disadvantage of the Goldstein conditions vis-à-vis the Wolfe conditions is that the first inequality in (3.11) may exclude all minimizers of ϕ. However, the Goldstein and Wolfe conditions have much in common, and their convergence theories are quite similar. The Goldstein conditions are often used in Newton-type methods but are not well suited for quasi-Newton methods that maintain a positive definite Hessian approximation.

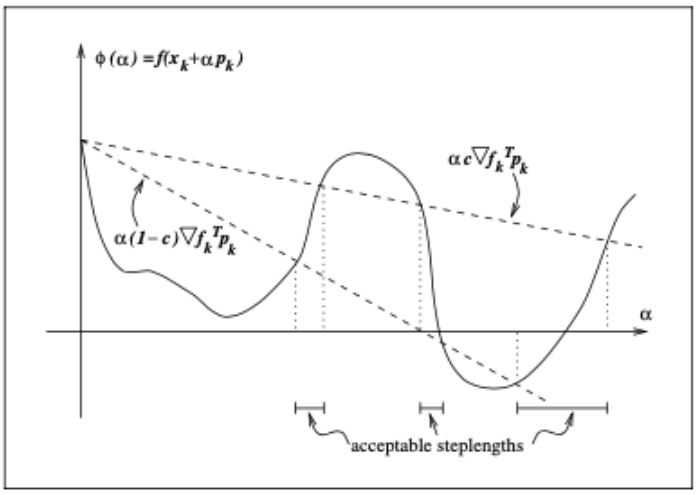

Figure 3.6 The Goldstein conditions.

Algorithm 3.1 (Backtracking Line Search).
Choose $\bar{\alpha} > 0$, $\rho \in (0, 1)$, $c \in (0, 1)$; Set $\alpha \leftarrow \bar{\alpha}$;
repeat until $f(x_k + \alpha p_k) \leq f(x_k) + c\alpha \nabla f_k^T p_k$
$\alpha \leftarrow \rho\alpha$;
end (repeat)
Terminate with $\alpha_k = \alpha$.

HESSIAN MODIFICATION

EIGENVALUE MODIFICATION

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

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

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

ADDING A MULTIPLE OF THE IDENTITY

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

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

4 - Trust-Region Methods

In this chapter, we will assume that the model function m_k that is used at each iterate x_k is quadratic. Moreover, m_k is based on the Taylor-series expansion of f around x_k, which is

$$f(x_k + p) = f_k + g_k^T p + \tfrac{1}{2} p^T \nabla^2 f(x_k + tp) p, \tag{4.1}$$

where $f_k = f(x_k)$ and $g_k = \nabla f(x_k)$, and t is some scalar in the interval $(0, 1)$. By using an approximation B_k to the Hessian in the second-order term, m_k is defined as follows:

$$m_k(p) = f_k + g_k^T p + \tfrac{1}{2} p^T B_k p, \tag{4.2}$$

where B_k is some symmetric matrix. The difference between $m_k(p)$ and $f(x_k + p)$ is $O\left(\|p\|^2\right)$, which is small when p is small.

When B_k is equal to the true Hessian $\nabla^2 f(x_k)$, the approximation error in the model function m_k is $O\left(\|p\|^3\right)$, so this model is especially accurate when $\|p\|$ is small. This choice $B_k = \nabla^2 f(x_k)$ leads to the trust-region Newton method, and will be discussed further in Section 4.4. In other sections of this chapter, we emphasize the generality of the trust-region approach by assuming little about B_k except symmetry and uniform boundedness.

To obtain each step, we seek a solution of the subproblem

$$\min_{p \in \mathbb{R}^n} m_k(p) = f_k + g_k^T p + \tfrac{1}{2} p^T B_k p \qquad \text{s.t. } \|p\| \le \Delta_k, \tag{4.3}$$

OUTLINE OF THE TRUST-REGION APPROACH

One of the key ingredients in a trust-region algorithm is the strategy for choosing the trust-region radius Δ_k at each iteration. We base this choice on the agreement between the model function m_k and the objective function f at previous iterations. Given a step p_k we define the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}; \tag{4.4}$$

the numerator is called the *actual reduction*, and the denominator is the *predicted reduction*

Algorithm 4.1 (Trust Region).
Given $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in \left[0, \frac{1}{4}\right)$:
for $k = 0, 1, 2, \ldots$
  Obtain p_k by (approximately) solving (4.3);
  Evaluate ρ_k from (4.4);
  if $\rho_k < \frac{1}{4}$
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  else
    if $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$
    else
      $\Delta_{k+1} = \Delta_k$;
  if $\rho_k > \eta$
    $x_{k+1} = x_k + p_k$
  else
    $x_{k+1} = x_k$;
end (for).

Theorem 4.1.

The vector p^* is a global solution of the trust-region problem

$$\min_{p\in\mathbb{R}^n} m(p) = f + g^T p + \tfrac{1}{2} p^T B p, \quad \text{s.t. } \|p\| \le \Delta, \tag{4.7}$$

if and only if p^* is feasible and there is a scalar $\lambda \ge 0$ such that the following conditions are satisfied:

$$(B + \lambda I)p^* = -g, \tag{4.8a}$$

$$\lambda(\Delta - \|p^*\|) = 0, \tag{4.8b}$$

$$(B + \lambda I) \quad \text{is positive semidefinite.} \tag{4.8c}$$

4.1 ALGORITHMS BASED ON THE CAUCHY POINT

Algorithm 4.2 (Cauchy Point Calculation).

Find the vector p_k^S that solves a linear version of (4.3), that is,

$$p_k^S = \arg\min_{p\in\mathbb{R}^n} f_k + g_k^T p \qquad \text{s.t. } \|p\| \le \Delta_k; \tag{4.9}$$

Calculate the scalar $\tau_k > 0$ that minimizes $m_k(\tau p_k^S)$ subject to satisfying the trust-region bound, that is,

$$\tau_k = \arg\min_{\tau\ge 0} m_k(\tau p_k^S) \qquad \text{s.t. } \|\tau p_k^S\| \le \Delta_k; \tag{4.10}$$

Set $p_k^C = \tau_k p_k^S$.

It is easy to write down a closed-form definition of the Cauchy point. For a start, the solution of (4.9) is simply

$$p_k^S = -\frac{\Delta_k}{\|g_k\|} g_k.$$

first. In summary, we have

$$p_k^C = -\tau_k \frac{\Delta_k}{\|g_k\|} g_k, \tag{4.11}$$

where

$$\tau_k = \begin{cases} 1 & \text{if } g_k^T B_k g_k \le 0; \\ \min\left(\|g_k\|^3/(\Delta_k g_k^T B_k g_k), 1\right) & \text{otherwise.} \end{cases} \tag{4.12}$$

THE DOGLEG METHOD

The first approach we discuss goes by the descriptive title of the *dogleg method*. It can be used when B is positive definite.

To motivate this method, we start by examining the effect of the trust-region radius Δ on the solution $p^*(\Delta)$ of the subproblem (4.5). When B is positive definite, we have already noted that the unconstrained minimizer of m is $p^B = -B^{-1}g$. When this point is feasible for (4.5), it is obviously a solution, so we have

$$p^*(\Delta) = p^B, \qquad \text{when } \Delta \ge \|p^B\|. \tag{4.13}$$

When Δ is small relative to p^B, the restriction $\|p\| \le \Delta$ ensures that the quadratic term in m has little effect on the solution of (4.5). For such Δ, we can get an approximation to $p(\Delta)$

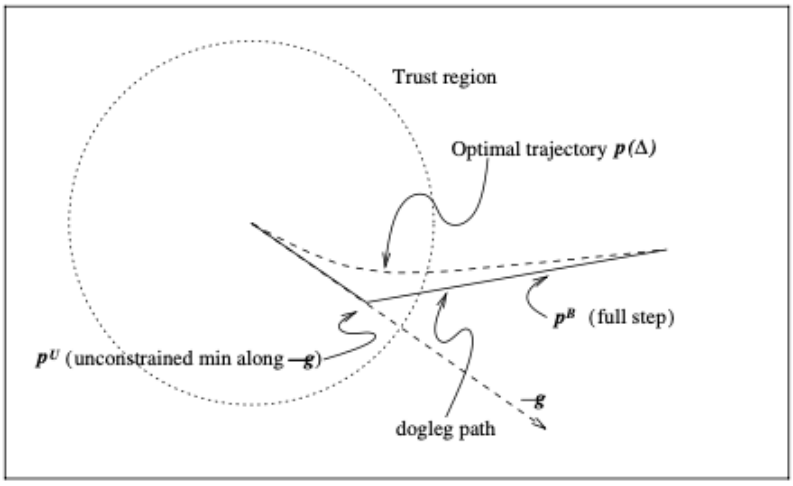

Figure 4.4 Exact trajectory and dogleg approximation.

by simply omitting the quadratic term from (4.5) and writing

$$p^*(\Delta) \approx -\Delta \frac{g}{\|g\|}, \qquad \text{when } \Delta \text{ is small.} \tag{4.14}$$

For intermediate values of Δ, the solution $p^*(\Delta)$ typically follows a curved trajectory like the one in Figure 4.4.

The dogleg method finds an approximate solution by replacing the curved trajectory for $p^*(\Delta)$ with a path consisting of two line segments. The first line segment runs from the origin to the minimizer of m along the steepest descent direction, which is

$$p^U = -\frac{g^T g}{g^T B g} g, \tag{4.15}$$

while the second line segment runs from p^U to p^B (see Figure 4.4). Formally, we denote this trajectory by $\tilde{p}(\tau)$ for $\tau \in [0, 2]$, where

$$\tilde{p}(\tau) = \begin{cases} \tau p^U, & 0 \le \tau \le 1, \\ p^U + (\tau - 1)(p^B - p^U), & 1 \le \tau \le 2. \end{cases} \tag{4.16}$$

The dogleg method chooses p to minimize the model m along this path, subject to the trust-region bound. The following lemma shows that the minimum along the dogleg path can be found easily.

4.3 ITERATIVE SOLUTION OF THE SUBPROBLEM

The characterization of Theorem 4.1 suggests an algorithm for finding the solution p of (4.7). Either $\lambda = 0$ satisfies (4.8a) and (4.8c) with $\|p\| \le \Delta$, or else we define

$$p(\lambda) = -(B + \lambda I)^{-1} g$$

for λ sufficiently large that $B + \lambda I$ is positive definite and seek a value $\lambda > 0$ such that

$$\|p(\lambda)\| = \Delta. \tag{4.37}$$

This problem is a one-dimensional root-finding problem in the variable λ.

To see that a value of λ with all the desired properties exists, we appeal to the eigendecomposition of B and use it to study the properties of $\|p(\lambda)\|$. Since B is symmetric, there is an orthogonal matrix Q and a diagonal matrix Λ such that $B = Q \Lambda Q^T$, where

$$\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n),$$

and $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the eigenvalues of B; see (A.16). Clearly, $B + \lambda I = Q(\Lambda + \lambda I)Q^T$, and for $\lambda \neq \lambda_j$, we have

$$p(\lambda) = -Q(\Lambda + \lambda I)^{-1} Q^T g = -\sum_{j=1}^{n} \frac{q_j^T g}{\lambda_j + \lambda} q_j, \tag{4.38}$$

where q_j denotes the jth column of Q. Therefore, by orthonormality of $q_1, q_2, \ldots, q_n$, we have

$$\|p(\lambda)\|^2 = \sum_{i=1}^{n} \frac{\left(q_j^T g\right)^2}{(\lambda_j + \lambda)^2}. \tag{4.39}$$

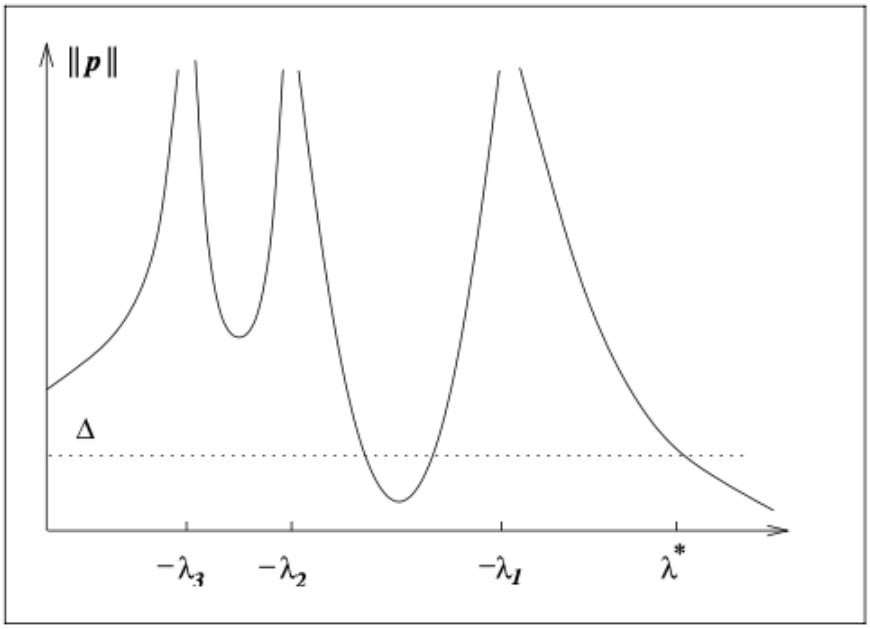

Figure 4.5 $\|p(\lambda)\|$ as a function of λ.

This expression tells us a lot about $\|p(\lambda)\|$. If $\lambda > -\lambda_1$, we have $\lambda_j + \lambda > 0$ for all $j = 1, 2, \ldots, n$, and so $\|p(\lambda)\|$ is a continuous, nonincreasing function of λ on the interval $(-\lambda_1, \infty)$. In fact, we have that

$$\lim_{\lambda \to \infty} \|p(\lambda)\| = 0. \tag{4.40}$$

Moreover, we have when $q_j^T g \neq 0$ that

$$\lim_{\lambda \to -\lambda_j} \|p(\lambda)\| = \infty. \tag{4.41}$$

Figure 4.5 plots $\|p(\lambda)\|$ against λ in a case in whcih $q_1^T g, q_2^T g$, and $q_3^T g$ are all nonzero. Note that the properties (4.40) and (4.41) hold and that $\|p(\lambda)\|$ is a nonincreasing function of λ on $(-\lambda_1, \infty)$. In particular, as is always the case when $q_1^T g \neq 0$, that there is a unique value $\lambda^* \in (-\lambda_1, \infty)$ such that $\|p(\lambda^*)\| = \Delta$. (There may be other, smaller values of λ for which $\|p(\lambda)\| = \Delta$, but these will fail to satisfy (4.8c).)

We now sketch a procedure for identifying the $\lambda^* \in (-\lambda_1, \infty)$ for which $\|p(\lambda^*)\| = \Delta$, which works when $q_1^T g \neq 0$. (We discuss the case of $q_1^T g = 0$ later.) First, note that when B positive definite and $\|B^{-1} g\| \leq \Delta$, the value $\lambda = 0$ satisfies (4.8), so the procedure can be terminated immediately with $\lambda^* = 0$. Otherwise, we could use the root-finding Newton's method (see the Appendix) to find the value of $\lambda > -\lambda_1$ that solves

$$\phi_1(\lambda) = \|p(\lambda)\| - \Delta = 0. \tag{4.42}$$

where $C_1 > 0$ and C_2 are constants. Clearly this approximation (and hence ϕ_1) is highly nonlinear, so the root-finding Newton's method will be unreliable or slow. Better results will be obtained if we reformulate the problem (4.42) so that it is nearly linear near the optimal λ. By defining

$$\phi_2(\lambda) = \frac{1}{\Delta} - \frac{1}{\|p(\lambda)\|},$$

it can be shown using (4.39) that for λ slightly greater than $-\lambda_1$, we have

$$\phi_2(\lambda) \approx \frac{1}{\Delta} - \frac{\lambda + \lambda_1}{C_3}$$

for some $C_3 > 0$. Hence, ϕ_2 is nearly linear near $-\lambda_1$ (see Figure 4.6), and the root-finding

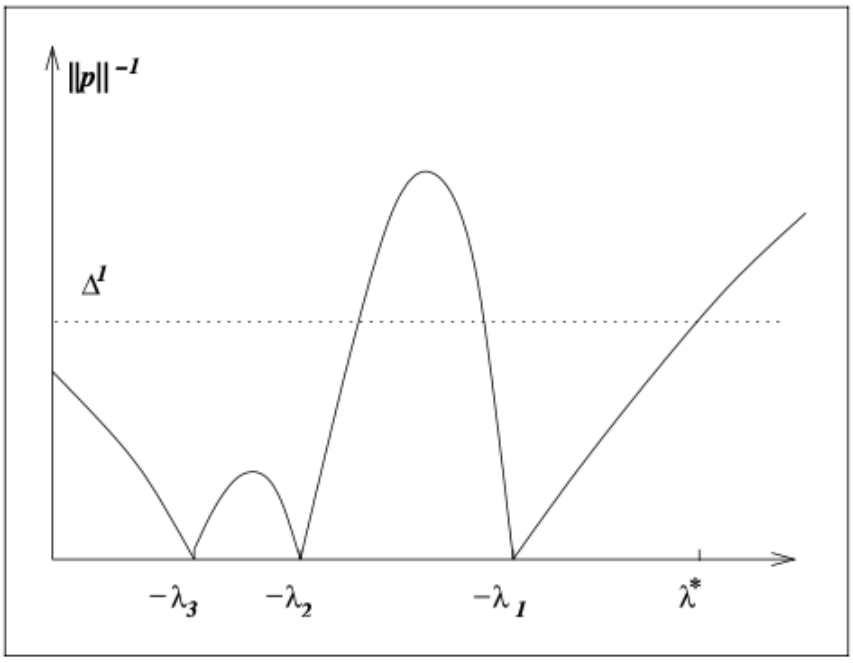

Figure 4.6 $1/\|p(\lambda)\|$ as a function of λ.

Newton's method will perform well, provided that it maintains $\lambda > -\lambda_1$. The root-finding Newton's method applied to ϕ_2 generates a sequence of iterates $\lambda^{(\ell)}$ by setting

$$\lambda^{(\ell+1)} = \lambda^{(\ell)} - \frac{\phi_2\left(\lambda^{(\ell)}\right)}{\phi_2'\left(\lambda^{(\ell)}\right)}. \tag{4.43}$$

After some elementary manipulation, this updating formula can be implemented in the following practical way.

Algorithm 4.3 (Trust Region Subproblem).
Given $\lambda^{(0)}$, $\Delta > 0$:
for $\ell = 0, 1, 2, \ldots$
Factor $B + \lambda^{(\ell)} I = R^T R$;
Solve $R^T R p_\ell = -g$, $R^T q_\ell = p_\ell$;

Set

$$\lambda^{(\ell+1)} = \lambda^{(\ell)} + \left(\frac{\|p_\ell\|}{\|q_\ell\|}\right)^2 \left(\frac{\|p_\ell\| - \Delta}{\Delta}\right); \tag{4.44}$$

end (for).

Safeguards must be added to this algorithm to make it practical; for instance, when $\lambda^{(\ell)} < -\lambda_1$, the Cholesky factorization $B + \lambda^{(\ell)} I = R^T R$ will not exist. A slightly enhanced version of this algorithm does, however, converge to a solution of (4.37) in most cases.

THE HARD CASE

Recall that in the discussion above, we assumed that $q_1^T g \neq 0$. In fact, the approach described above can be applied even when the most negative eigenvalue is a multiple eigenvalue (that is, $0 > \lambda_1 = \lambda_2 = \cdots$), provided that $Q_1^T g \neq 0$, where Q_1 is the matrix whose columns span the subspace corresponding to the eigenvalue λ_1. When this condition does not hold, the situation becomes a little complicated, because the limit (4.41) does not hold for $\lambda_j = \lambda_1$ and so there may not be a value $\lambda \in (-\lambda_1, \infty)$ such that $\|p(\lambda)\| = \Delta$ (see Figure 4.7). Moré and Sorensen [214] refer to this case as the *hard case*. At first glance, it is not clear how p and λ can be chosen to satisfy (4.8) in the hard case. Clearly, our root-finding technique will not work, since there is no solution for λ in the open interval $(-\lambda_1, \infty)$. But Theorem 4.1 assures us that the right value of λ lies in the interval $[-\lambda_1, \infty)$, so there is only

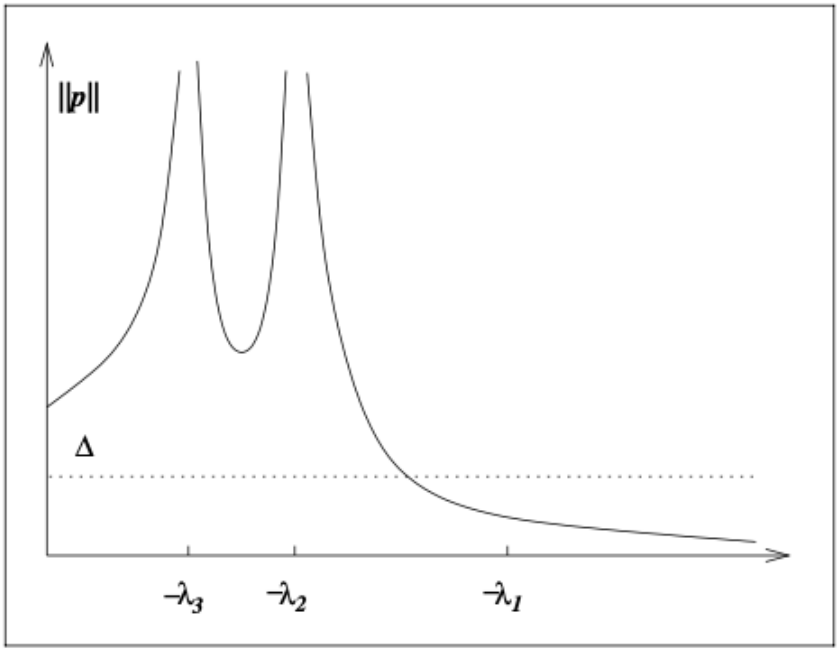

Figure 4.7 The hard case: $\|p(\lambda)\| < \Delta$ for all $\lambda \in (-\lambda_1, \infty)$.

one possibility: $\lambda = -\lambda_1$. To find p, it is not enough to delete the terms for which $\lambda_j = \lambda_1$ from the formula (4.38) and set

$$p = \sum_{j:\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j + \lambda} q_j.$$

Instead, we note that $(B - \lambda_1 I)$ is singular, so there is a vector z such that $\|z\| = 1$ and $(B - \lambda_1 I)z = 0$. In fact, z is an eigenvector of B corresponding to the eigenvalue λ_1, so by orthogonality of Q we have $q_j^T z = 0$ for $\lambda_j \neq \lambda_1$. It follows from this property that if we set

$$p = \sum_{j:\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j + \lambda} q_j + \tau z \tag{4.45}$$

for any scalar τ, we have

$$\|p\|^2 = \sum_{j:\lambda_j \neq \lambda_1} \frac{\left(q_j^T g\right)^2}{(\lambda_j + \lambda)^2} + \tau^2,$$

so it is always possible to choose τ to ensure that $\|p\| = \Delta$. It is easy to check that the conditions (4.8) holds for this choice of p and $\lambda = -\lambda_1$.

4.4 LOCAL CONVERGENCE OF TRUST-REGION NEWTON METHODS

Reasonable implementations of the dogleg, subspace minimization, and nearly-exact algorithm of Section 4.3 with $B_k = \nabla^2 f(x_k)$ eventually use the steps $p_k = p_k^N$ under the conditions of Theorem 4.9, and therefore converge quadratically. In the case of the dogleg and two-dimensional subspace minimization methods, the exact step p_k^N is one of the candidates for p_k—it lies inside the trust region, along the dogleg path, and inside the two-dimensional subspace. Since under the assumptions of Theorem 4.9, p_k^N is the unconstrained minimizer of m_k for k sufficiently large, it is certainly the minimizer in the more restricted domains, so we have $p_k = p_k^N$. For the approach of Section 4.3, if we follow the reasonable strategy of checking whether p_k^N is a solution of (4.3) prior to embarking on Algorithm 4.3, then eventually we will also have $p_k = p_k^N$ also.

5 - Quasi-Newton Methods

6.1 THE BFGS METHOD

The most popular quasi-Newton algorithm is the BFGS method, named for its discoverers Broyden, Fletcher, Goldfarb, and Shanno. In this section we derive this algorithm (and its close relative, the DFP algorithm) and describe its theoretical properties and practical implementation.

We begin the derivation by forming the following quadratic model of the objective function at the current iterate x_k:

$$m_k(p) = f_k + \nabla f_k^T p + \tfrac{1}{2} p^T B_k p. \tag{6.1}$$

Here B_k is an $n \times n$ symmetric positive definite matrix that will be revised or *updated* at every iteration. Note that the function value and gradient of this model at $p = 0$ match f_k and ∇f_k, respectively. The minimizer p_k of this convex quadratic model, which we can write explicitly as

$$p_k = -B_k^{-1} \nabla f_k, \tag{6.2}$$

is used as the search direction, and the new iterate is

$$x_{k+1} = x_k + \alpha_k p_k, \tag{6.3}$$

where the step length α_k is chosen to satisfy the Wolfe conditions (3.6).

What requirements should we impose on B_{k+1}, based on the knowledge gained during the latest step? One reasonable requirement is that the gradient of m_{k+1} should match the gradient of the objective function f at the latest two iterates x_k and x_{k+1}. Since $\nabla m_{k+1}(0)$ is precisely ∇f_{k+1}, the second of these conditions is satisfied automatically. The first condition can be written mathematically as

$$\nabla m_{k+1}(-\alpha_k p_k) = \nabla f_{k+1} - \alpha_k B_{k+1} p_k = \nabla f_k.$$

By rearranging, we obtain

$$B_{k+1} \alpha_k p_k = \nabla f_{k+1} - \nabla f_k. \tag{6.4}$$

To simplify the notation it is useful to define the vectors

$$s_k = x_{k+1} - x_k = \alpha_k p_k, \qquad y_k = \nabla f_{k+1} - \nabla f_k, \tag{6.5}$$

so that (6.4) becomes

$$B_{k+1} s_k = y_k. \tag{6.6}$$

We refer to this formula as the *secant equation.*

Given the displacement s_k and the change of gradients y_k, the secant equation requires that the symmetric positive definite matrix B_{k+1} map s_k into y_k. This will be possible only if s_k and y_k satisfy the *curvature condition*

$$s_k^T y_k > 0, \tag{6.7}$$

To determine B_{k+1} uniquely, we impose the additional condition that *among all symmetric matrices satisfying the secant equation, B_{k+1} is, in some sense, closest to the current matrix B_k.* In other words, we solve the problem

$$\min_B \|B - B_k\| \tag{6.9a}$$

$$\text{subject to} \quad B = B^T, \quad Bs_k = y_k, \tag{6.9b}$$

A norm that allows easy solution of the minimization problem (6.9) and gives rise to a scale-invariant optimization method is the weighted Frobenius norm

$$\|A\|_W \equiv \left\|W^{1/2} A W^{1/2}\right\|_F, \tag{6.10}$$

where $\|\cdot\|_F$ is defined by $\|C\|_F^2 = \sum_{i=1}^n \sum_{j=1}^n c_{ij}^2$. The weight matrix W can be chosen as *any* matrix satisfying the relation $W y_k = s_k$. For concreteness, the reader can assume that $W = \bar{G}_k^{-1}$ where $\bar{G}_k$ is the *average Hessian* defined by

DFP Update

With this weighting matrix and this norm, the unique solution of (6.9) is

$$\text{(DFP)} \qquad B_{k+1} = \left(I - \rho_k y_k s_k^T\right) B_k \left(I - \rho_k s_k y_k^T\right) + \rho_k y_k y_k^T, \tag{6.13}$$

with

$$\rho_k = \frac{1}{y_k^T s_k}. \tag{6.14}$$

to be calculated by means of a simple matrix–vector multiplication. Using the Sherman–Morrison–Woodbury formula (A.28), we can derive the following expression for the update of the inverse Hessian approximation H_k that corresponds to the DFP update of B_k in (6.13):

$$\text{(DFP)} \qquad H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{y_k^T s_k}. \tag{6.15}$$

BFGS Update

$$\text{(BFGS)} \qquad B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}. \tag{6.19}$$

$$\text{(BFGS)} \qquad H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \tag{6.17}$$

Algorithm 6.1 (BFGS Method).

Given starting point x_0, convergence tolerance $\epsilon > 0$,
    inverse Hessian approximation H_0;
$k \leftarrow 0$;
while $\|\nabla f_k\| > \epsilon$;
    Compute search direction

$$p_k = -H_k \nabla f_k; \tag{6.18}$$

    Set $x_{k+1} = x_k + \alpha_k p_k$ where α_k is computed from a line search
        procedure to satisfy the Wolfe conditions (3.6);
    Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$;
    Compute H_{k+1} by means of (6.17);
    $k \leftarrow k + 1$;
end (while)

Each iteration can be performed at a cost of $O(n^2)$ arithmetic operations (plus the cost of function and gradient evaluations); there are no $O(n^3)$ operations such as linear system solves or matrix–matrix operations. The algorithm is robust, and its rate of convergence is superlinear, which is fast enough for most practical purposes. Even though Newton's method converges more rapidly (that is, quadratically), its cost per iteration usually is higher, because of its need for second derivatives and solution of a linear system.

6.2 THE SR1 METHOD

In the BFGS and DFP updating formulae, the updated matrix B_{k+1} (or H_{k+1}) differs from its predecessor B_k (or H_k) by a rank-2 matrix. In fact, as we now show, there is a simpler rank-1 update that maintains symmetry of the matrix and allows it to satisfy the secant equation. Unlike the rank-two update formulae, this *symmetric-rank-1*, or SR1, update does not guarantee that the updated matrix maintains positive definiteness. Good numerical results have been obtained with algorithms based on SR1, so we derive it here and investigate its properties.

The symmetric rank-1 update has the general form

$$B_{k+1} = B_k + \sigma v v^T,$$

where σ is either $+1$ or -1, and σ and v are chosen so that B_{k+1} satisfies the secant equation (6.6), that is, $y_k = B_{k+1}s_k$. By substituting into this equation, we obtain

$$y_k = B_k s_k + \left[\sigma v^T s_k\right] v. \tag{6.22}$$

Since the term in brackets is a scalar, we deduce that v must be a multiple of $y_k - B_k s_k$, that is, $v = \delta(y_k - B_k s_k)$ for some scalar δ. By substituting this form of v into (6.22), we obtain

$$(y_k - B_k s_k) = \sigma\delta^2 \left[s_k^T (y_k - B_k s_k)\right](y_k - B_k s_k), \tag{6.23}$$

and it is clear that this equation is satisfied if (and only if) we choose the parameters δ and σ to be

$$\sigma = \operatorname{sign}\left[s_k^T (y_k - B_k s_k)\right], \qquad \delta = \pm \left|s_k^T (y_k - B_k s_k)\right|^{-1/2}.$$

Hence, we have shown that the only symmetric rank-1 updating formula that satisfies the secant equation is given by

$$\text{(SR1)} \qquad B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}. \tag{6.24}$$

By applying the Sherman–Morrison formula (A.27), we obtain the corresponding update formula for the inverse Hessian approximation H_k:

$$\text{(SR1)} \qquad H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k}. \tag{6.25}$$

The main drawback of SR1 updating is that the denominator in (6.24) or (6.25) can vanish. In fact, even when the objective function is a convex quadratic, there may be steps on which there is no symmetric rank-1 update that satisfies the secant equation. It pays to reexamine the derivation above in the light of this observation.

By reasoning in terms of B_k (similar arguments can be applied to H_k), we see that there are three cases:

1. If $(y_k - B_k s_k)^T s_k \neq 0$, then the arguments above show that there is a unique rank-one updating formula satisfying the secant equation (6.6), and that it is given by (6.24).

2. If $y_k = B_k s_k$, then the only updating formula satisfying the secant equation is simply $B_{k+1} = B_k$.

3. If $y_k \neq B_k s_k$ and $(y_k - B_k s_k)^T s_k = 0$, then (6.23) shows that there is no symmetric rank-one updating formula satisfying the secant equation.

We now introduce a strategy to prevent the SR1 method from breaking down. It has been observed in practice that SR1 performs well simply by skipping the update if the denominator is small. More specifically, the update (6.24) is applied only if

$$\left|s_k^T(y_k - B_k s_k)\right| \geq r\|s_k\| \, \|y_k - B_k s_k\|, \tag{6.26}$$

where $r \in (0, 1)$ is a small number, say $r = 10^{-8}$. If (6.26) does not hold, we set $B_{k+1} = B_k$. Most implementations of the SR1 method use a skipping rule of this kind.

Why do we advocate skipping of updates for the SR1 method, when in the previous section we discouraged this strategy in the case of BFGS? The two cases are quite different. The condition $s_k^T(y_k - B_k s_k) \approx 0$ occurs infrequently, since it requires certain vectors to be aligned in a specific way. When it does occur, skipping the update appears to have no negative effects on the iteration. This is not surprising, since the skipping condition implies that $s_k^T \bar{G} s_k \approx s_k^T B_k s_k$, where $\bar{G}$ is the average Hessian over the last step—meaning that the curvature of B_k along s_k is already correct. In contrast, the curvature condition $s_k^T y_k \geq 0$ required for BFGS updating may easily fail if the line search does not impose the Wolfe conditions (for example, if the step is not long enough), and therefore skipping the BFGS update can occur often and can degrade the quality of the Hessian approximation.

Algorithm 6.2 (SR1 Trust-Region Method).
Given starting point x_0, initial Hessian approximation B_0,
trust-region radius Δ_0, convergence tolerance $\epsilon > 0$,
parameters $\eta \in (0, 10^{-3})$ and $r \in (0, 1)$;
$k \leftarrow 0$;
while $\|\nabla f_k\| > \epsilon$;
Compute s_k by solving the subproblem

$$\min_s \nabla f_k^T s + \frac{1}{2} s^T B_k s \qquad \text{subject to } \|s\| \leq \Delta_k; \tag{6.27}$$

Compute

$$\begin{aligned} y_k &= \nabla f(x_k + s_k) - \nabla f_k, \\ \text{ared} &= f_k - f(x_k + s_k) \qquad \text{(actual reduction)} \\ \text{pred} &= -\left(\nabla f_k^T s_k + \frac{1}{2} s_k^T B_k s_k\right) \qquad \text{(predicted reduction);} \end{aligned}$$

if ared/pred $> \eta$
$x_{k+1} = x_k + s_k$;
else
$x_{k+1} = x_k$;
end (if)

if ared/pred > 0.75
if $\|s_k\| \leq 0.8\Delta_k$
$\Delta_{k+1} = \Delta_k$;
else
$\Delta_{k+1} = 2\Delta_k$;
end (if)
else if $0.1 \leq$ ared/pred ≤ 0.75
$\Delta_{k+1} = \Delta_k$;
else
$\Delta_{k+1} = 0.5\Delta_k$;
end (if)
if (6.26) holds
Use (6.24) to compute B_{k+1} (even if $x_{k+1} = x_k$);
else
$B_{k+1} \leftarrow B_k$;
end (if)
$k \leftarrow k + 1$;
end (while)

6 - Constrained Optimization

$$\min_{x \in \mathbb{R}^n} f(x) \qquad \text{subject to} \quad \begin{cases} c_i(x) = 0, & i \in \mathcal{E}, \\ c_i(x) \geq 0, & i \in \mathcal{I}, \end{cases} \tag{12.1}$$

where f and the functions c_i are all smooth, real-valued functions on a subset of $\mathbb{R}^n$, and $\mathcal{I}$ and $\mathcal{E}$ are two finite sets of indices. As before, we call f the *objective* function, while c_i, $i \in \mathcal{E}$ are the *equality constraints* and c_i, $i \in \mathcal{I}$ are the *inequality constraints.*

Necessary conditions: Local unconstrained minimizers have $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ positive semidefinite.

Sufficient conditions: Any point x^* at which $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite is a strong local minimizer of f.

A vector x^* is a *local solution* of the problem (12.3) if $x^* \in \Omega$ and there is a neighborhood $\mathcal{N}$ of x^* such that $f(x) \geq f(x^*)$ for $x \in \mathcal{N} \cap \Omega$.

Similarly, we can make the following definitions:

A vector x^* is a *strict local solution* (also called a *strong local solution*) if $x^* \in \Omega$ and there is a neighborhood $\mathcal{N}$ of x^* such that $f(x) > f(x^*)$ for all $x \in \mathcal{N} \cap \Omega$ with $x \neq x^*$.

A point x^* is an *isolated local solution* if $x^* \in \Omega$ and there is a neighborhood $\mathcal{N}$ of x^* such that x^* is the only local solution in $\mathcal{N} \cap \Omega$.

Definition 12.1.

The active set $\mathcal{A}(x)$ *at any feasible x consists of the equality constraint indices from $\mathcal{E}$ together with the indices of the inequality constraints i for which $c_i(x) = 0$; that is,*

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(x) = 0\}.$$

At a feasible point x, the inequality constraint $i \in \mathcal{I}$ is said to be *active* if $c_i(x) = 0$ and *inactive* if the strict inequality $c_i(x) > 0$ is satisfied.

Definition 12.4 (LICQ).

Given the point x and the active set $\mathcal{A}(x)$ defined in Definition 12.1, we say that the linear independence constraint qualification (LICQ) *holds if the set of active constraint gradients $\{\nabla c_i(x), i \in \mathcal{A}(x)\}$ is linearly independent.*

12.3 FIRST-ORDER OPTIMALITY CONDITIONS

As a preliminary to stating the necessary conditions, we define the Lagrangian function for the general problem (12.1).

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x). \tag{12.33}$$

Theorem 12.1 (First-Order Necessary Conditions).

Suppose that x^* is a local solution of (12.1), that the functions f and c_i in (12.1) are continuously differentiable, and that the LICQ holds at x^*. Then there is a Lagrange multiplier vector λ^*, with components λ_i^*, $i \in \mathcal{E} \cup \mathcal{I}$, such that the following conditions are satisfied at (x^*, λ^*)

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = 0, \tag{12.34a}$$
$$c_i(x^*) = 0, \quad \text{for all } i \in \mathcal{E}, \tag{12.34b}$$
$$c_i(x^*) \geq 0, \quad \text{for all } i \in \mathcal{I}, \tag{12.34c}$$
$$\lambda_i^* \geq 0, \quad \text{for all } i \in \mathcal{I}, \tag{12.34d}$$
$$\lambda_i^* c_i(x^*) = 0, \quad \text{for all } i \in \mathcal{E} \cup \mathcal{I}. \tag{12.34e}$$

The conditions (12.34) are often known as the *Karush–Kuhn–Tucker conditions*, or *KKT conditions* for short. The conditions (12.34e) are *complementarity conditions*; they imply that either constraint i is active or $\lambda_i^* = 0$, or possibly both. In particular, the Lagrange multipliers corresponding to inactive inequality constraints are zero, we can omit the terms for indices $i \notin \mathcal{A}(x^*)$ from (12.34a) and rewrite this condition as

$$0 = \nabla_x \mathcal{L}(x^*, \lambda^*) = \nabla f(x^*) - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* \nabla c_i(x^*). \tag{12.35}$$

Definition 12.5 (Strict Complementarity).

Given a local solution x^* of (12.1) and a vector λ^* satisfying (12.34), we say that the strict complementarity condition *holds if* exactly one *of λ_i^* and $c_i(x^*)$ is zero for each index $i \in \mathcal{I}$. In other words, we have that $\lambda_i^* > 0$ for each $i \in \mathcal{I} \cap \mathcal{A}(x^*)$.*

Satisfaction of the strict complementarity property usually makes it easier for algorithms to determine the active set $\mathcal{A}(x^*)$ and converge rapidly to the solution x^*.

For a given problem (12.1) and solution point x^*, there may be many vectors λ^* for which the conditions (12.34) are satisfied. When the LICQ holds, however, the optimal λ^* is unique (see Exercise 12.17).