CHAPTER 11

Nonlinear Equations

In many applications we do not need to optimize an objective function explicitly, but rather to find values of the variables in a model that satisfy a number of given relationships. When these relationships take the form of n equalities—the same number of equality conditions as variables in the model—the problem is one of solving a system of *nonlinear equations*. We write this problem mathematically as

$$r(x) = 0, \tag{11.1}$$

where $r : \mathbb{R}^n \to \mathbb{R}^n$ is a vector function, that is,

$$r(x) = \begin{bmatrix} r_1(x) \\ r_2(x) \\ \vdots \\ r_n(x) \end{bmatrix}.$$

In this chapter, we assume that each function $r_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2, \ldots, n$, is smooth. A vector x^* for which (11.1) is satisfied is called a *solution* or *root* of the nonlinear equations. A simple example is the system

$$r(x) = \begin{bmatrix} x_2^2 - 1 \\ \sin x_1 - x_2 \end{bmatrix} = 0,$$

which is a system of $n = 2$ equations with infinitely many solutions, two of which are $x^* = (3\pi/2, -1)^T$ and $x^* = (\pi/2, 1)^T$. In general, the system (11.1) may have no solutions, a unique solution, or many solutions.

The techniques for solving nonlinear equations overlap in their motivation, analysis, and implementation with optimization techniques discussed in earlier chapters. In both optimization and nonlinear equations, Newton's method lies at the heart of many important algorithms. Features such as line searches, trust regions, and inexact solution of the linear algebra subproblems at each iteration are important in both areas, as are other issues such as derivative evaluation and global convergence.

Because some important algorithms for nonlinear equations proceed by minimizing a sum of squares of the equations, that is,

$$\min_x \sum_{i=1}^{n} r_i^2(x),$$

there are particularly close connections with the nonlinear least-squares problem discussed in Chapter 10. The differences are that in nonlinear equations, the number of equations *equals* the number of variables (instead of exceeding the number of variables, as is typically the case in Chapter 10), and that we expect all equations to be satisfied at the solution, rather than just minimizing the sum of squares. This point is important because the nonlinear equations may represent physical or economic constraints such as conservation laws or consistency principles, which must hold exactly in order for the solution to be meaningful.

Many applications require us to solve a sequence of closely related nonlinear systems, as in the following example.

❑ **EXAMPLE 11.1** (RHEINBOLDT; SEE [212])

An interesting problem in control is to analyze the stability of an aircraft in response to the commands of the pilot. The following is a simplified model based on force-balance equations, in which gravity terms have been neglected.

The equilibrium equations for a particular aircraft are given by a system of 5 equations in 8 unknowns of the form

$$F(x) \equiv Ax + \phi(x) = 0, \tag{11.2}$$

where $F : \mathbb{R}^8 \to \mathbb{R}^5$, the matrix A is given by

$$A = \begin{bmatrix} -3.933 & 0.107 & 0.126 & 0 & -9.99 & 0 & -45.83 & -7.64 \\ 0 & -0.987 & 0 & -22.95 & 0 & -28.37 & 0 & 0 \\ 0.002 & 0 & -0.235 & 0 & 5.67 & 0 & -0.921 & -6.51 \\ 0 & 1.0 & 0 & -1.0 & 0 & -0.168 & 0 & 0 \\ 0 & 0 & -1.0 & 0 & -0.196 & 0 & -0.0071 & 0 \end{bmatrix},$$

and the nonlinear part is defined by

$$\phi(x) = \begin{bmatrix} -0.727x_2x_3 + 8.39x_3x_4 - 684.4x_4x_5 + 63.5x_4x_2 \\ 0.949x_1x_3 + 0.173x_1x_5 \\ -0.716x_1x_2 - 1.578x_1x_4 + 1.132x_4x_2 \\ -x_1x_5 \\ x_1x_4 \end{bmatrix}.$$

The first three variables x_1, x_2, x_3, represent the rates of roll, pitch, and yaw, respectively, while x_4 is the incremental angle of attack and x_5 the sideslip angle. The last three variables x_6, x_7, x_8 are the controls; they represent the deflections of the elevator, aileron, and rudder, respectively.

For a given choice of the control variables x_6, x_7, x_8 we obtain a system of 5 equations and 5 unknowns. If we wish to study the behavior of the aircraft as the controls are changed, we need to solve a system of nonlinear equations with unknowns $x_1, x_2, \ldots, x_5$ for each setting of the controls.

❑

Despite the many similarities between nonlinear equations and unconstrained and least-squares optimization algorithms, there are also some important differences. To obtain quadratic convergence in optimization we require second derivatives of the objective function, whereas knowledge of the first derivatives is sufficient in nonlinear equations.

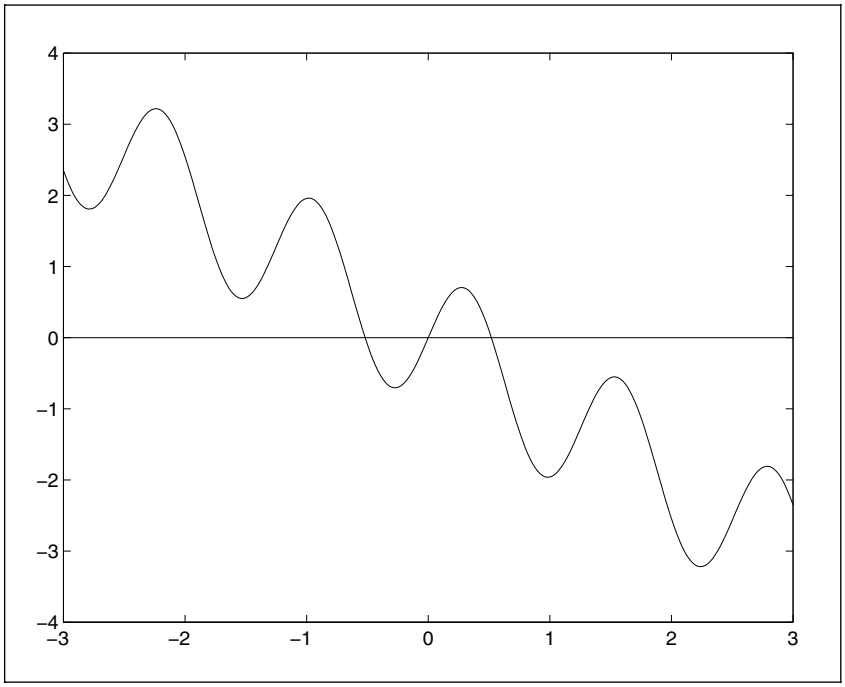

Figure 11.1 The function $r(x) = \sin(5x) - x$ has three roots.

Quasi-Newton methods are perhaps less useful in nonlinear equations than in optimization. In unconstrained optimization, the objective function is the natural choice of merit function that gauges progress towards the solution, but in nonlinear equations various merit functions can be used, all of which have some drawbacks. Line search and trust-region techniques play an equally important role in optimization, but one can argue that trust-region algorithms have certain theoretical advantages in solving nonlinear equations.

Some of the difficulties that arise in trying to solve nonlinear equations can be illustrated by a simple scalar example ($n = 1$). Suppose we have

$$r(x) = \sin(5x) - x, \tag{11.3}$$

as plotted in Figure 11.1. From this figure we see that there are three solutions of the problem $r(x) = 0$, also known as *roots of* r, located at zero and approximately ± 0.519148. This situation of multiple solutions is similar to optimization problems where, for example, a function may have more than one local minimum. It is not *quite* the same, however: In the case of optimization, one of the local minima may have a lower function value than the others (making it a "better" solution), while in nonlinear equations all solutions are equally good from a mathematical viewpoint. (If the modeler decides that the solution

found by the algorithm makes no sense on physical grounds, their model may need to be reformulated.)

In this chapter we start by outlining algorithms related to Newton's method and examining their local convergence properties. Besides Newton's method itself, these include Broyden's quasi-Newton method, inexact Newton methods, and tensor methods. We then address global convergence, which is the issue of trying to force convergence to a solution from a remote starting point. Finally, we discuss a class of methods in which an "easy" problem—one to which the solution is well known—is gradually transformed into the problem $F(x) = 0$. In these so-called continuation (or homotopy) methods, we track the solution as the problem changes, with the aim of finishing up at a solution of $F(x) = 0$.

Throughout this chapter we make the assumption that the vector function r is continuously differentiable in the region $\mathcal{D}$ containing the values of x we are interested in. In other words, the Jacobian $J(x)$ (the matrix of first partial derivatives of $r(x)$ defined in the Appendix and in (10.3)) exists and is continuous. We say that x^* satisfying $r(x^*) = 0$ is a *degenerate solution* if $J(x^*)$ is singular, and a *nondegenerate solution* otherwise.

11.1 LOCAL ALGORITHMS

NEWTON'S METHOD FOR NONLINEAR EQUATIONS

Recall from Theorem 2.1 that Newton's method for minimizing $f : \mathbb{R}^n \to \mathbb{R}$ forms a quadratic model function by taking the first three terms of the Taylor series approximation of f around the current iterate x_k. The Newton step is the vector that minimizes this model. In the case of nonlinear equations, Newton's method is derived in a similar way, but with a *linear* model, one that involves function values and first derivatives of the functions $r_i(x)$, $i = 1, 2, \ldots, m$ at the current iterate x_k. We justify this strategy by referring to the following multidimensional variant of Taylor's theorem.

Theorem 11.1.

Suppose that $r : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable in some convex open set $\mathcal{D}$ and that x and $x + p$ are vectors in $\mathcal{D}$. We then have that

$$r(x + p) = r(x) + \int_0^1 J(x + tp)p \, dt. \tag{11.4}$$

We can define a linear model $M_k(p)$ of $r(x_k + p)$ by approximating the second term on the right-hand-side of (11.4) by $J(x)p$, and writing

$$M_k(p) \stackrel{\text{def}}{=} r(x_k) + J(x_k)p. \tag{11.5}$$

Newton's method, in its pure form, chooses the step p_k to be the vector for which $M_k(p_k) = 0$, that is, $p_k = -J(x_k)^{-1}r(x_k)$. We define it formally as follows.

Algorithm 11.1 (Newton's Method for Nonlinear Equations).
Choose x_0;
for $k = 0, 1, 2, \ldots$
Calculate a solution p_k to the Newton equations

$$J(x_k)p_k = -r(x_k); \tag{11.6}$$

$x_{k+1} \leftarrow x_k + p_k$;
end (for)

We use a linear model to derive the Newton step, rather than a quadratic model as in unconstrained optimization, because the linear model normally has a solution and yields an algorithm with rapid convergence properties. In fact, Newton's method for unconstrained optimization (see (2.15)) can be derived by applying Algorithm 11.1 to the nonlinear equations $\nabla f(x) = 0$. We see also in Chapter 18 that sequential quadratic programming for equality-constrained optimization can be derived by applying Algorithm 11.1 to the nonlinear equations formed by the first-order optimality conditions (18.3) for this problem. Another connection is with the Gauss–Newton method for nonlinear least squares; the formula (11.6) is equivalent to (10.23) in the usual case in which $J(x_k)$ is nonsingular.

When the iterate x_k is close to a nondegenerate root x^*, Newton's method converges superlinearly, as we show in Theorem 11.2 below. Potential shortcomings of the method include the following.

- When the starting point is remote from a solution, Algorithm 11.1 can behave erratically. When $J(x_k)$ is singular, the Newton step may not even be defined.
- First-derivative information (the Jacobian matrix J) may be difficult to obtain.
- It may be too expensive to find and calculate the Newton step p_k exactly when n is large.
- The root x^* in question may be degenerate, that is, $J(x^*)$ may be singular.

An example of a degenerate problem is the scalar function $r(x) = x^2$, which has a single degenerate root at $x^* = 0$. Algorithm 11.1, when started from any nonzero x_0, generates the sequence of iterates

$$x_k = \frac{1}{2^k}x_0,$$

which converges to the solution 0, but only at a linear rate.

As we show later in this chapter, Newton's method can be modified and enhanced in various ways to get around most of these problems. The variants we describe form the basis of much of the available software for solving nonlinear equations.

We summarize the local convergence properties of Algorithm 11.1 in the following theorem. For part of this result, we make use of a Lipschitz continuity assumption on the Jacobian, by which we mean that there is a constant β_L such that

$$\|J(x_0) - J(x_1)\| \leq \beta_L \|x_0 - x_1\|, \tag{11.7}$$

for all x_0 and x_1 in the domain in question.

Theorem 11.2.

Suppose that r is continuously differentiable in a convex open set $\mathcal{D} \subset \mathbb{R}^n$. Let $x^* \in \mathcal{D}$ be a nondegenerate solution of $r(x) = 0$, and let $\{x_k\}$ be the sequence of iterates generated by Algorithm 11.1. Then when $x_k \in \mathcal{D}$ is sufficiently close to x^*, we have

$$x_{k+1} - x^* = o(\|x_k - x^*\|), \tag{11.8}$$

indicating local Q-superlinear convergence. When r is Lipschitz *continuously differentiable near x^*, we have for all x_k sufficiently close to x^* that*

$$x_{k+1} - x^* = O(\|x_k - x^*\|^2), \tag{11.9}$$

indicating local Q-quadratic convergence.

PROOF. Since $r(x^*) = 0$, we have from Theorem 11.1 that

$$r(x_k) = r(x_k) - r(x^*) = J(x_k)(x_k - x^*) + w(x_k, x^*), \tag{11.10}$$

where

$$w(x_k, x^*) = \int_0^1 \left[J(x_k + t(x^* - x_k)) - J(x_k)\right](x_k - x^*). \tag{11.11}$$

From (A.12) and continuity of J, we have

$$\begin{aligned} \|w(x_k, x^*)\| &= \left\| \int_0^1 [J(x^* + t(x^* - x_k)) - J(x_k)](x_k - x^*)\, dt \right\| \\ &\leq \int_0^1 \|J(x^* + t(x^* - x_k)) - J(x_k)\|\, \|x_k - x^*\|\, dt \\ &= o(\|x_k - x^*\|). \end{aligned} \tag{11.12}$$

Since $J(x^*)$ is nonsingular, there is a radius $\delta > 0$ and a positive constant β^* such that for all x in the ball $\mathcal{B}(x^*, \delta)$ defined by

$$\mathcal{B}(x^*, \delta) = \{x \mid \|x - x^*\| \leq \delta\}, \tag{11.13}$$

we have that

$$\|J(x)^{-1}\| \leq \beta^* \quad \text{and} \quad x \in \mathcal{D}. \tag{11.14}$$

Assuming that $x_k \in \mathcal{B}(x^*, \delta)$, and recalling the definition (11.6), we multiply both sides of (11.10) by $J(x_k)^{-1}$ to obtain

$$\begin{aligned} -p_k &= (x_k - x^*) + \|J(x_k)^{-1}\| o(\|x_k - x^*\|), \\ \Rightarrow \quad x_k + p_k - x^* &= o(\|x_k - x^*\|), \\ \Rightarrow \quad x_{k+1} - x^* &= o(\|x_k - x^*\|), \end{aligned} \tag{11.15}$$

which yields (11.8).

When the Lipschitz continuity assumption (11.7) is satisfied, we can obtain a sharper estimate for the remainder term $w(x_k, x^*)$ defined in (11.11). By using (11.7) in (11.12), we obtain

$$\|w(x_k, x^*)\| = O(\|x_k - x^*\|^2). \tag{11.16}$$

By multiplying (11.10) by $J(x_k)^{-1}$ as above, we obtain

$$-p_k - (x_k - x^*) = J(x_k)^{-1} w(x_k, x^*),$$

so the estimate (11.9) follows as in (11.15). □

INEXACT NEWTON METHODS

Instead of solving (11.6) exactly, inexact Newton methods use search directions p_k that satisfy the condition

$$\|r_k + J_k p_k\| \leq \eta_k \|r_k\|, \quad \text{for some } \eta_k \in [0, \eta], \tag{11.17}$$

where $\eta \in [0, 1)$ is a constant. As in Chapter 7, we refer to $\{\eta_k\}$ as the *forcing sequence*. Different methods make different choices of the forcing sequence, and they use different algorithms for finding the approximate solutions p_k. The general framework for this class of methods can be stated as follows.

Framework 11.2 (Inexact Newton for Nonlinear Equations).
Given $\eta \in [0, 1)$;
Choose x_0;
for $k = 0, 1, 2, \ldots$
    Choose forcing parameter $\eta_k \in [0, \eta]$;
    Find a vector p_k that satisfies (11.17);
    $x_{k+1} \leftarrow x_k + p_k$;
end (for)

The convergence theory for these methods depends only on the condition (11.17) and not on the particular technique used to calculate p_k. The most important methods in this class, however, make use of iterative techniques for solving linear systems of the form $Jp = -r$, such as GMRES (Saad and Schultz [273], Walker [302]) or other Krylov-space methods. Like the conjugate-gradient algorithm of Chapter 5 (which is not directly applicable here, since the coefficient matrix J is not symmetric positive definite), these methods typically require us to perform a matrix–vector multiplication of the form Jd for some d at each iteration, and to store a number of work vectors of length n. GMRES requires an additional vector to be stored at each iteration, so must be restarted periodically (often every 10 or 20 iterations) to keep memory requirements at a reasonable level.

The matrix–vector products Jd can be computed without explicit knowledge of the Jacobian J. A finite-difference approximation to Jd that requires one evaluation of $r(\cdot)$ is given by the formula (8.11). Calculation of Jd exactly (at least, to within the limits of finite-precision arithmetic) can be performed by using the forward mode of automatic differentiation, at a cost of at most a small multiple of an evaluation of $r(\cdot)$. Details of this procedure are given in Section 8.2.

We do not discuss the iterative methods for sparse linear systems here, but refer the interested reader to Kelley [177] and Saad [272] for comprehensive descriptions and implementations of the most interesting techniques. We prove a local convergence theorem for the method, similar to Theorem 11.2.

Theorem 11.3.

Suppose that r is continuously differentiable in a convex open set $\mathcal{D} \subset \mathbb{R}^n$. Let $x^* \in \mathcal{D}$ be a nondegenerate solution of $r(x) = 0$, and let $\{x_k\}$ be the sequence of iterates generated by the Framework 11.2. Then when $x_k \in \mathcal{D}$ is sufficiently close to x^*, the following are true:

(i) *If η in (11.17) is sufficiently small, the convergence of $\{x_k\}$ to x^* is Q-linear.*

(ii) *If $\eta_k \to 0$, the convergence is Q-superlinear.*

(iii) *If, in addition, $J(\cdot)$ is Lipschitz continuous in a neighborhood of x^* and $\eta_k = O(\|r_k\|)$, the convergence is Q-quadratic.*

PROOF. We first rewrite (11.17) as

$$J(x_k)p_k + r(x_k) = v_k, \quad \text{where } \|v_k\| \le \eta_k \|r(x_k)\|. \tag{11.18}$$

Since x^* is a nondegenerate root, we have as in (11.14) that there is a radius $\delta > 0$ such that $\|J(x)^{-1}\| \le \beta^*$ for some constant β^* and all $x \in \mathcal{B}(x^*, \delta)$. By multiplying both sides of (11.18) by $J(x_k)^{-1}$ and rearranging, we find that

$$\left\| p_k + J(x_k)^{-1} r(x_k) \right\| = \left\| J(x_k)^{-1} v_k \right\| \le \beta^* \eta_k \|r(x_k)\|. \tag{11.19}$$

As in (11.10), we have that

$$r(x) = J(x)(x - x^*) + w(x, x^*), \tag{11.20}$$

where $\rho(x) \stackrel{\text{def}}{=} \|w(x, x^*)\|/\|x - x^*\| \to 0$ as $x \to x^*$. By reducing δ if necessary, we have from this expression that the following bound holds for all $x \in \mathcal{B}(x^*, \delta)$:

$$\|r(x)\| \leq 2\|J(x^*)\| \, \|x - x^*\| + o(\|x - x^*\|) \leq 4\|J(x^*)\| \, \|x - x^*\|. \tag{11.21}$$

We now set $x = x_k$ in (11.20), and use (11.19) and (11.21) to obtain

$$\begin{aligned} \|x_k + p_k - x^*\| &= \left\| p_k + J(x_k)^{-1}(r(x_k) - w(x_k, x^*)) \right\| \\ &\leq \beta^* \eta_k \|r(x_k)\| + \|J(x_k)^{-1}\| \|w(x_k, x^*)\| \\ &\leq \left[4\|J(x^*)\|\beta^* \eta_k + \beta^* \rho(x_k) \right] \|x_k - x^*\|. \end{aligned} \tag{11.22}$$

By choosing x_k close enough to x^* that $\rho(x_k) \leq 1/(4\beta^*)$, and choosing $\eta = 1/(8\|J(x^*)\|\beta^*)$, we have that the term in square brackets in (11.22) is at most $1/2$. Hence, since $x_{k+1} = x_k + p_k$, this formula indicates Q-linear convergence of $\{x_k\}$ to x^*, proving part (i).

Part (ii) follows immediately from the fact that the term in brackets in (11.22) goes to zero as $x_k \to x^*$ and $\eta_k \to 0$. For part (iii), we combine the techniques above with the logic of the second part of the proof of Theorem 11.2. Details are left as an exercise. □

BROYDEN'S METHOD

Secant methods, also known as quasi-Newton methods, do not require calculation of the Jacobian $J(x)$. Instead, they construct their own approximation to this matrix, updating it at each iteration so that it mimics the behavior of the true Jacobian J over the step just taken. The approximate Jacobian, which we denote at iteration k by B_k, is then used to construct a linear model analogous to (11.5), namely

$$M_k(p) = r(x_k) + B_k p. \tag{11.23}$$

We obtain the step by setting this model to zero. When B_k is nonsingular, we have the following explicit formula (cf. (11.6)):

$$p_k = -B_k^{-1} r(x_k). \tag{11.24}$$

The requirement that the approximate Jacobian should mimic the behavior of the true Jacobian can be specified as follows. Let s_k denote the step from x_k to x_{k+1}, and let y_k

be the corresponding change in r, that is,

$$s_k = x_{k+1} - x_k, \qquad y_k = r(x_{k+1}) - r(x_k). \tag{11.25}$$

From Theorem 11.1, we have that s_k and y_k are related by the expression

$$y_k = \int_0^1 J(x_k + ts_k)s_k \, dt \approx J(x_{k+1})s_k + o(\|s_k\|). \tag{11.26}$$

We require the updated Jacobian approximation B_{k+1} to satisfy the following equation, which is known as the *secant equation*,

$$y_k = B_{k+1}s_k, \tag{11.27}$$

which ensures that B_{k+1} and $J(x_{k+1})$ have similar behavior along the direction s_k. (Note the similarity with the secant equation (6.6) in quasi-Newton methods for unconstrained optimization; the motivation is the same in both cases.) The secant equation does not say anything about how B_{k+1} should behave along directions orthogonal to s_k. In fact, we can view (11.27) as a system of n linear equations in n^2 unknowns, where the unknowns are the components of B_{k+1}, so for $n > 1$ the equation (11.27) does not determine all the components of B_{k+1} uniquely. (The scalar case of $n = 1$ gives rise to the scalar secant method; see (A.60).)

The most successful practical algorithm is Broyden's method, for which the update formula is

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)s_k^T}{s_k^T s_k}. \tag{11.28}$$

The Broyden update makes the smallest possible change to the Jacobian (as measured by the Euclidean norm $\|B_k - B_{k+1}\|_2$) that is consistent with (11.27), as we show in the following Lemma.

Lemma 11.4 (Dennis and Schnabel [92, Lemma 8.1.1]).

Among all matrices B satisfying $Bs_k = y_k$, the matrix B_{k+1} defined by (11.28) minimizes the difference $\|B - B_k\|$.

PROOF. Let B be any matrix that satisfies $Bs_k = y_k$. By the properties of the Euclidean norm (see (A.10)) and the fact that $\|ss^T/s^Ts\| = 1$ for any vector s (see Exercise 11.1), we have

$$\begin{aligned}\|B_{k+1} - B_k\| &= \left\| \frac{(y_k - B_k s_k)s_k^T}{s_k^T s_k} \right\| \\ &= \left\| \frac{(B - B_k)s_k s_k^T}{s_k^T s_k} \right\| \leq \|B - B_k\| \left\| \frac{s_k s_k^T}{s_k^T s_k} \right\| = \|B - B_k\|.\end{aligned}$$

Hence, we have that

$$B_{k+1} \in \arg \min_{B\,:\,y_k = B s_k} \|B - B_k\|,$$

and the result is proved. □

In the specification of the algorithm below, we allow a line search to be performed along the search direction p_k, so that $s_k = \alpha p_k$ for some $\alpha > 0$ in the formula (11.25). (See below for details about line-search methods.)

Algorithm 11.3 (Broyden).
Choose x_0 and a nonsingular initial Jacobian approximation B_0;
for $k = 0, 1, 2, \ldots$

Calculate a solution p_k to the linear equations

$$B_k p_k = -r(x_k); \tag{11.29}$$

Choose α_k by performing a line search along p_k;
$x_{k+1} \leftarrow x_k + \alpha_k p_k$;
$s_k \leftarrow x_{k+1} - x_k$;
$y_k \leftarrow r(x_{k+1}) - r(x_k)$;
Obtain B_{k+1} from the formula (11.28);
end (for)

Under certain assumptions, Broyden's method converges *superlinearly*, that is,

$$\|x_{k+1} - x^*\| = o(\|x_k - x^*\|). \tag{11.30}$$

This local convergence rate is fast enough for most practical purposes, though not as fast as the Q-quadratic convergence of Newton's method.

We illustrate the difference between the convergence rates of Newton's and Broyden's method with a small example. The function $r : \mathbb{R}^2 \to \mathbb{R}^2$ defined by

$$r(x) = \begin{bmatrix} (x_1 + 3)(x_2^3 - 7) + 18 \\ \sin(x_2 e^{x_1} - 1) \end{bmatrix} \tag{11.31}$$

has a nondegenerate root at $x^* = (0, 1)^T$. We start both methods from the point $x_0 = (-0.5, 1.4)^T$, and use the exact Jacobian $J(x_0)$ at this point as the initial Jacobian approximation B_0. Results are shown in Table 11.1.

Newton's method clearly exhibits Q-quadratic convergence, which is characterized by doubling of the exponent of the error at each iteration. Broyden's method takes twice as

Table 11.1 Convergence of Iterates in Broyden and Newton Methods

	$\|x_k - x^*\|_2$	
Iteration k	Broyden	Newton
0	0.64×10^{0}	0.64×10^{0}
1	0.62×10^{-1}	0.62×10^{-1}
2	0.52×10^{-3}	0.21×10^{-3}
3	0.25×10^{-3}	0.18×10^{-7}
4	0.43×10^{-4}	0.12×10^{-15}
5	0.14×10^{-6}	
6	0.57×10^{-9}	
7	0.18×10^{-11}	
8	0.87×10^{-15}	

Table 11.2 Convergence of Function Norms in Broyden and Newton Methods

	$\|r(x_k)\|_2$	
Iteration k	Broyden	Newton
0	0.74×10^{1}	0.74×10^{1}
1	0.59×10^{0}	0.59×10^{0}
2	0.20×10^{-2}	0.23×10^{-2}
3	0.21×10^{-2}	0.16×10^{-6}
4	0.37×10^{-3}	0.22×10^{-15}
5	0.12×10^{-5}	
6	0.49×10^{-8}	
7	0.15×10^{-10}	
8	0.11×10^{-18}	

many iterations as Newton's, and reduces the error at a rate that accelerates slightly towards the end. The function norms $\|r(x_k)\|$ approach zero at a similar rate to the iteration errors $\|x_k - x^*\|$. As in (11.10), we have that

$$r(x_k) = r(x_k) - r(x^*) \approx J(x^*)(x_k - x^*),$$

so by nonsingularity of $J(x^*)$, the norms of $r(x_k)$ and $(x_k - x^*)$ are bounded above and below by multiples of each other. For our example problem (11.31), convergence of the sequence of function norms in the two methods is shown in Table 11.2.

The convergence analysis of Broyden's method is more complicated than that of Newton's method. We state the following result without proof.

Theorem 11.5.

Suppose the assumptions of Theorem 11.2 hold. Then there are positive constants ϵ and δ such that if the starting point x_0 and the starting approximate Jacobian B_0 satisfy

$$\|x_0 - x^*\| \le \delta, \quad \|B_0 - J(x^*)\| \le \epsilon, \tag{11.32}$$

the sequence $\{x_k\}$ generated by Broyden's method (11.24), (11.28) is well-defined and converges Q-superlinearly to x^*.

The second condition in (11.32)—that the initial Jacobian approximation B_0 must be close to the true Jacobian at the solution $J(x^*)$—is difficult to guarantee in practice. In contrast to the case of unconstrained minimization, a good choice of B_0 can be critical to the performance of the algorithm. Some implementations of Broyden's method recommend choosing B_0 to be $J(x_0)$, or some finite-difference approximation to this matrix.

The Broyden matrix B_k will be dense in general, even if the true Jacobian J is sparse. Therefore, when n is large, an implementation of Broyden's method that stores B_k as a full $n \times n$ matrix may be inefficient. Instead, we can use limited-memory methods in which B_k is stored implicitly in the form of a number of vectors of length n, while the system (11.29) is solved by a technique based on application of the Sherman–Morrison–Woodbury formula (A.28). These methods are similar to the ones described in Chapter 7 for large-scale unconstrained optimization.

TENSOR METHODS

In tensor methods, the linear model $M_k(p)$ used by Newton's method (11.5) is augmented with an extra term that aims to capture some of the nonlinear, higher-order, behavior of r. By doing so, it achieves more rapid and reliable convergence to degenerate roots, in particular, to roots x^* for which the Jacobian $J(x^*)$ has rank $n-1$ or $n-2$. We give a broad outline of the method here, and refer to Schnabel and Frank [277] for details.

We use $\hat{M}_k(p)$ to denote the model function on which tensor methods are based; this function has the form

$$\hat{M}_k(p) = r(x_k) + J(x_k)p + \tfrac{1}{2}T_k pp, \tag{11.33}$$

where T_k is a tensor defined by n^3 elements $(T_k)_{ijl}$ whose action on a pair of arbitrary vectors u and v in $\mathbb{R}^n$ is defined by

$$(T_k uv)_i = \sum_{j=1}^{n}\sum_{l=1}^{n}(T_k)_{ijl}u_j v_l.$$

If we followed the reasoning behind Newton's method, we could consider building T_k from the *second* derivatives of r at the point x_k, that is,

$$(T_k)_{ijl} = [\nabla^2 r_i(x_k)]_{jl}.$$

For instance, in the example (11.31), we have that

$$(T(x)uv)_1 = u^T \nabla^2 r_1(x) v = u^T \begin{bmatrix} 0 & 3x_2^2 \\ 3x_2^2 & 6x_2(x_1+3) \end{bmatrix} v$$
$$= 3x_2^2(u_1v_2 + u_2v_1) + 6x_2(x_1+3)u_2v_2.$$

However, use of the exact second derivatives is not practical in most instances. If we were to store this information explicitly, about $n^3/2$ memory locations would be needed, about n times the requirements of Newton's method. Moreover, there may be no vector p for which $\hat{M}_k(p) = 0$, so the step may not even be defined.

Instead, the approach described in [277] defines T_k in a way that requires little additional storage, but which gives $\hat{M}_k$ some potentially appealing properties. Specifically, T_k is chosen so that $\hat{M}_k(p)$ interpolates the function $r(x_k + p)$ at some previous iterates visited by the algorithm. That is, we require that

$$\hat{M}_k(x_{k-j} - x_k) = r(x_{k-j}), \quad \text{for } j = 1, 2, \ldots, q, \tag{11.34}$$

for some integer $q > 0$. By substituting from (11.33), we see that T_k must satisfy the condition

$$\tfrac{1}{2} T_k s_{jk} s_{jk} = r(x_{k-j}) - r(x_k) - J(x_k) s_{jk},$$

where

$$s_{jk} \stackrel{\text{def}}{=} x_{k-j} - x_k, \ j = 1, 2, \ldots, q.$$

In [277] it is shown that this condition can be ensured by choosing T_k so that its action on arbitrary vectors u and v is

$$T_k uv = \sum_{j=1}^{q} a_j (s_{jk}^T u)(s_{jk}^T v),$$

where a_j, $j = 1, 2, \ldots, q$, are vectors of length n. The number of interpolating points q is typically chosen to be quite modest, usually less than $\sqrt{n}$. This T_k can be stored in $2nq$ locations, which contain the vectors a_j and s_{jk} for $j = 1, 2, \ldots, q$. Note the connection between this idea and Broyden's method, which also chooses information in the model (albeit in the *first-order* part of the model) to interpolate the function value at the previous iterate.

This technique can be refined in various ways. The points of interpolation can be chosen to make the collection of directions s_{jk} more linearly independent. There may still not be a vector p for which $\hat{M}_k(p) = 0$, but we can instead take the step to be the vector that

minimizes $\|\hat{M}_k(p)\|_2^2$, which can be found by using a specialized least-squares technique. There is no assurance that the step obtained in this way is a descent direction for the merit function $\frac{1}{2}\|r(x)\|^2$ (which is discussed in the next section), and in this case it can be replaced by the standard Newton direction $-J_k^{-1}r_k$.

11.2 PRACTICAL METHODS

We now consider practical variants of the Newton-like methods discussed above, in which line-search and trust-region modifications to the steps are made in order to ensure better global convergence behavior.

MERIT FUNCTIONS

As mentioned above, neither Newton's method (11.6) nor Broyden's method (11.24), (11.28) with unit step lengths can be guaranteed to converge to a solution of $r(x) = 0$ unless they are started close to that solution. Sometimes, components of the unknown or function vector or the Jacobian will blow up. Another, more exotic, kind of behavior is *cycling*, where the iterates move between distinct regions of the parameter space without approaching a root. An example is the scalar function

$$r(x) = -x^5 + x^3 + 4x,$$

which has five nondegenerate roots. When started from the point $x_0 = 1$, Newton's method produces a sequence of iterates that oscillates between 1 and -1 (see Exercise 11.3) without converging to any of the roots.

The Newton and Broyden methods can be made more robust by using line-search and trust-region techniques similar to those described in Chapters 3 and 4. Before describing these techniques, we need to define a *merit function*, which is a scalar-valued function of x that indicates whether a new iterate is better or worse than the current iterate, in the sense of making progress toward a root of r. In unconstrained optimization, the objective function f is itself a natural merit function; most algorithms for minimizing f require a decrease in f at each iteration. In nonlinear equations, the merit function is obtained by combining the n components of the vector r in some way.

The most widely used merit function is the sum of squares, defined by

$$f(x) = \tfrac{1}{2}\|r(x)\|^2 = \tfrac{1}{2}\sum_{i=1}^{n} r_i^2(x). \tag{11.35}$$

(The factor $1/2$ is introduced for convenience.) Any root x^* of r obviously has $f(x^*) = 0$, and since $f(x) \geq 0$ for all x, each root is a minimizer of f. However, local minimizers of f are not roots of r if f is strictly positive at the point in question. Still, the merit function

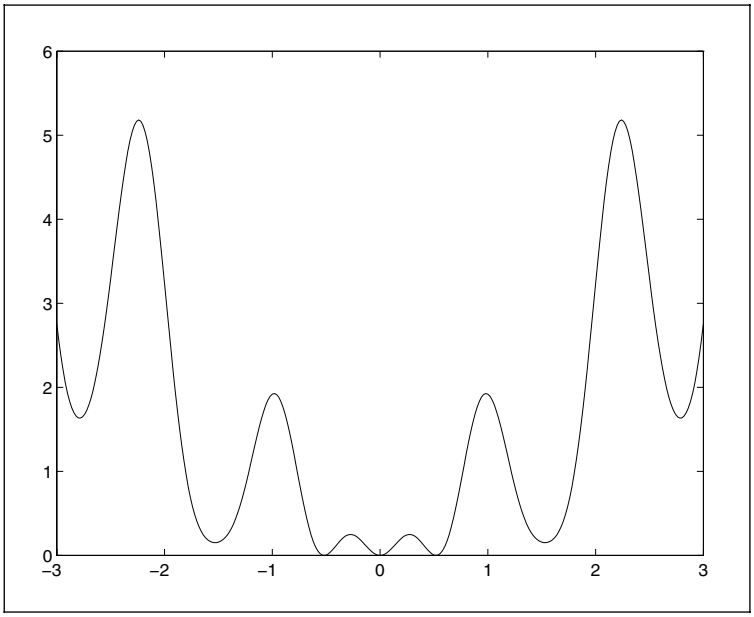

Figure 11.2 Plot of $\frac{1}{2}[\sin(5x) - x]^2$, showing its many local minima.

(11.35) has been used successfully in many applications and is implemented in a number of software packages.

The merit function for the example (11.3) is plotted in Figure 11.2. It shows three local minima corresponding to the three roots, but there are many other local minima (for example, those at around ± 1.53053). Local minima like these that are not roots of f satisfy an interesting property. Since

$$\nabla f(x^*) = J(x^*)^T r(x^*) = 0, \tag{11.36}$$

we can have $r(x^*) \neq 0$ only if $J(x^*)$ is singular.

Since local minima for the sum-of-squares merit function may be points of attraction for the algorithms described in this section, global convergence results for the algorithms discussed here are less satisfactory than for similar algorithms applied to unconstrained optimization.

Other merit functions are also used in practice. One such is the ℓ_1 norm merit function defined by

$$f_1(x) = \|r(x)\|_1 = \sum_{i=1}^{m} |r_i(x)|.$$

This function is studied in Chapters 17 and 18 in the context of algorithms for constrained optimization.

LINE SEARCH METHODS

We can obtain algorithms with global convergence properties by applying the line-search approach of Chapter 3 to the sum-of-squares merit function $f(x) = \frac{1}{2}\|r(x)\|^2$. When it is well defined, the Newton step

$$J(x_k)p_k = -r(x_k) \tag{11.37}$$

is a descent direction for $f(\cdot)$ whenever $r_k \neq 0$, since

$$p_k^T \nabla f(x_k) = -p_k^T J_k^T r_k = -\|r_k\|^2 < 0. \tag{11.38}$$

Step lengths α_k are chosen by one of the procedures of Chapter 3, and the iterates are defined by the formula

$$x_{k+1} = x_k + \alpha_k p_k, \qquad k = 0, 1, 2, \ldots. \tag{11.39}$$

For the case of line searches that choose α_k to satisfy the Wolfe conditions (3.6), we have the following convergence result, which follows directly from Theorem 3.2.

Theorem 11.6.

Suppose that $J(x)$ is Lipschitz continuous in a neighborhood $\mathcal{D}$ of the level set $\mathcal{L} = \{x : f(x) \leq f(x_0)\}$, and that $\|J(x)\|$ and $\|r(x)\|$ are bounded above on $\mathcal{D}$. Suppose that a line-search algorithm (11.39) is applied to f, where the search directions p_k satisfy $p_k^T \nabla f_k < 0$ while the step lengths α_k satisfy the Wolfe conditions (3.6). Then we have that the Zoutendijk condition holds, that is,

$$\sum_{k \geq 0} \cos^2 \theta_k \|J_k^T r_k\|^2 < \infty,$$

where

$$\cos \theta_k = \frac{-p_k^T \nabla f(x_k)}{\|p_k\| \|\nabla f(x_k)\|}. \tag{11.40}$$

We omit the proof, which verifies that ∇f is Lipschitz continuous on $\mathcal{D}$ and that f is bounded below (by 0) on $\mathcal{D}$, and then applies Theorem 3.2.

Provided that the sequence of iterates satisfies

$$\cos \theta_k \geq \delta, \qquad \text{for some } \delta \in (0, 1) \text{ and all } k \text{ sufficiently large,} \tag{11.41}$$

Theorem 11.6 guarantees that $J_k^T r_k \to 0$, meaning that the iterates approach stationarity of the merit function f. Moreover, if we know that $\|J(x_k)^{-1}\|$ is bounded then we must have $r_k \to 0$.

We now investigate the values of $\cos\theta_k$ for the directions generated by the Newton and inexact Newton methods. From (11.40) and (11.38), we have for the exact Newton step (11.6) that

$$\cos\theta_k = -\frac{p_k^T \nabla f(x_k)}{\|p_k\|\|\nabla f(x_k)\|} = \frac{\|r_k\|^2}{\|J_k^{-1} r_k\|\|J_k^T r_k\|} \geq \frac{1}{\|J_k^T\|\|J_k^{-1}\|} = \frac{1}{\kappa(J_k)}. \tag{11.42}$$

When p_k is an inexact Newton direction—that is, one that satisfies the condition (11.17)—we have that

$$\begin{aligned} \|r_k + J_k p_k\|^2 \leq \eta_k^2 \|r_k\|^2 &\Rightarrow 2p_k^T J_k^T r_k + \|r_k\|^2 + \|J_k p_k\|^2 \leq \eta^2 \|r_k\|^2 \\ &\Rightarrow p_k^T \nabla f_k = p_k^T J_k^T r_k \leq [(\eta^2 - 1)/2]\|r_k\|^2. \end{aligned}$$

Meanwhile,

$$\|p_k\| \leq \|J_k^{-1}\| \, [\|r_k + J_k p_k\| + \|r_k\|] \leq \|J_k^{-1}\|(\eta + 1)\|r_k\|,$$

and

$$\|\nabla f_k\| = \|J_k^T r_k\| \leq \|J_k\|\|r_k\|.$$

By combining these estimates, we obtain

$$\cos\theta_k = -\frac{p_k^T \nabla f_k}{\|p_k\|\|\nabla f_k\|} \geq \frac{1 - \eta^2}{2\|J_k\|\|J_k^{-1}\|(1 + \eta)} \geq \frac{1 - \eta}{2\kappa(J_k)}.$$

We conclude that a bound of the form (11.41) is satisfied both for the exact and inexact Newton methods, provided that the condition number $\kappa(J_k)$ is bounded.

When $\kappa(J_k)$ is large, however, this lower bound is close to zero, and use of the Newton direction may cause poor performance of the algorithm. In fact, the following example shows that condition $\cos\theta_k$ can converge to zero, causing the algorithm to fail. This example highlights a fundamental weakness of the line-search approach.

❑ **EXAMPLE 11.2** (POWELL [241])

Consider the problem of finding a solution of the nonlinear system

$$r(x) = \begin{bmatrix} x_1 \\ \dfrac{10x_1}{(x_1 + 0.1)} + 2x_2^2 \end{bmatrix}, \tag{11.43}$$

with unique solution $x^* = 0$. We try to solve this problem using the Newton iteration (11.37), (11.39) where α_k is chosen to minimize f along p_k. It is proved in [241] that, starting from the point $(3, 1)^T$, the iterates converge to $(1.8016, 0)^T$ (to four digits of accuracy). However, this point is not a solution of (11.43). In fact, it is not even a stationary point of f, and a step from this point in the direction $-\nabla f$ will produce a decrease in both components of r. To verify these claims, note that the Jacobian of r, which is

$$J(x) = \begin{bmatrix} 1 & 0 \\ \dfrac{1}{(x_1 + 0.1)^2} & 4x_2 \end{bmatrix},$$

is singular at all x for which $x_2 = 0$. For such points, we have

$$\nabla f(x) = \begin{bmatrix} x_1 + \dfrac{10x_1}{(x_1 + 0.1)^3} \\ 0 \end{bmatrix},$$

so that the gradient points in the direction of the positive x_1 axis whenever $x_1 > 0$. The point $(1.8016, 0)^T$ is therefore not a stationary point of f.

For this example, a calculation shows that the Newton step generated from an iterate that is close to (but not quite on) the x_1 axis tends to be parallel to the x_2 axis, making it nearly orthogonal to the gradient $\nabla f(x)$. That is, $\cos\theta_k$ for the Newton direction may be arbitrarily close to zero.

❐

In this example, a Newton method with exact line searches is attracted to a point of no interest at which the Jacobian is singular. Since systems of nonlinear equations often contain singular points, this behavior gives cause for concern.

To prevent this undesirable behavior and ensure that (11.41) holds, we may have to modify the Newton direction. One possibility is to add some multiple $\lambda_k I$ of the identity to $J_k^T J_k$, and define the step p_k to be

$$p_k = -(J_k^T J_k + \lambda_k I)^{-1} J_k^T r_k. \tag{11.44}$$

For any $\lambda_k > 0$ the matrix in parentheses is nonsingular, and if λ_k is bounded away from zero, a condition of the form (11.41) is satisfied. Therefore, some practical algorithms choose λ_k adaptively to ensure that the matrix in (11.44) does not approach singularity. This approach is analogous to the classical Levenberg-Marquardt algorithm discussed in Chapter 10. To implement it without forming $J_k^T J_k$ explicitly and performing trial Cholesky factorizations of the matrices $(J_k^T J_k + \lambda I)$, we can use the technique (10.36) illustrated earlier for the least-squares case. This technique uses the fact that the Cholesky factor of $(J_k^T J_k + \lambda I)$ is

identical to R^T, where R is the upper triangular factor from the QR factorization of the matrix

$$\begin{bmatrix} J_k \\ \sqrt{\lambda} I \end{bmatrix}. \tag{11.45}$$

A combination of Householder and Givens transformations can be used, as for (10.36), and the savings noted in the discussion following (10.36) continue to hold if we need to perform this calculation for several candidate values of λ_k.

The drawback of this Levenberg-Marquardt approach is that it is difficult to choose λ_k. If too large, we can destroy the fast rate of convergence of Newton's method. (Note that p_k approaches a multiple of $-J_k^T r_k$ as $\lambda_k \uparrow \infty$, so the step becomes small and tends to point in the steepest-descent direction for f.) If λ_k is too small, the algorithm can be inefficient in the presence of Jacobian singularities. A more satisfactory approach is to follow the trust-region approach described below, which chooses λ_k indirectly.

We conclude by specifying an algorithm based on Newton-like steps and line searches that regularizes the step calculations where necessary. Several details are deliberately left vague; we refer the reader to the papers cited above for details.

Algorithm 11.4 (Line Search Newton-like Method).
Given c_1, c_2 with $0 < c_1 < c_2 < \frac{1}{2}$;
Choose x_0;
for $k = 0, 1, 2, \ldots$
    Calculate a Newton-like step from (11.6) (regularizing with (11.44)
        if J_k appears to be near-singular), or (11.17) or (11.24);
    if $\alpha = 1$ satisfies the Wolfe conditions (3.6)
        Set $\alpha_k = 1$;
    else
        Perform a line search to find $\alpha_k > 0$ that satisfies (3.6);
    end (if)
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
end (for)

TRUST-REGION METHODS

The most widely used trust-region methods for nonlinear equations simply apply Algorithm 4.1 from Chapter 4 to the merit function $f(x) = \frac{1}{2}\|r(x)\|_2^2$, using $B_k = J(x_k)^T J(x_k)$ as the approximate Hessian in the model function $m_k(p)$, which is defined as follows:

$$m_k(p) = \tfrac{1}{2}\|r_k + J_k p\|_2^2 = f_k + p^T J_k^T r_k + \tfrac{1}{2} p^T J_k^T J_k p_k.$$

The step p_k is generated by finding an approximate solution of the subproblem

$$\min_p m_k(p), \qquad \text{subject to } \|p\| \le \Delta_k, \tag{11.46}$$

where Δ_k is the radius of the trust region. The ratio ρ_k of actual to predicted reduction (see (4.4)), which plays a critical role in many trust-region algorithms, is therefore

$$\rho_k = \frac{\|r(x_k)\|^2 - \|r(x_k + p_k)\|^2}{\|r(x_k)\|^2 - \|r(x_k) + J(x_k)p_k\|^2}. \tag{11.47}$$

We can state the trust-region framework that results from this model as follows.

Algorithm 11.5 (Trust-Region Method for Nonlinear Equations).
Given $\bar{\Delta} > 0$, $\Delta_0 \in (0, \bar{\Delta})$, and $\eta \in [0, \frac{1}{4})$:
for $k = 0, 1, 2, \ldots$
  Calculate p_k as an (approximate) solution of (11.46);
  Evaluate ρ_k from (11.47);
  if $\rho_k < \frac{1}{4}$
    $\Delta_{k+1} = \frac{1}{4}\|p_k\|$;
  else
    if $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min(2\Delta_k, \bar{\Delta})$;
    else
      $\Delta_{k+1} = \Delta_k$;
    end (if)
  end (if)
  if $\rho_k > \eta$
    $x_{k+1} = x_k + p_k$;
  else
    $x_{k+1} = x_k$;
  end (if)
end (for).

The dogleg method is a special case of the trust-region algorithm, Algorithm 4.1, that constructs an approximate solution to (11.46) based on the Cauchy point p_k^c and the unconstrained minimizer of m_k. The Cauchy point is

$$p_k^c = -\tau_k(\Delta_k / \|J_k^T r_k\|) J_k^T r_k, \tag{11.48}$$

where

$$\tau_k = \min\left\{1, \|J_k^T r_k\|^3 / (\Delta_k r_k^T J_k (J_k^T J_k) J_k^T r_k)\right\}; \tag{11.49}$$

By comparing with the general definition (4.11), (4.12) we see that it is not necessary to consider the case of an indefinite Hessian approximation in $m_k(p)$, since the model Hessian $J_k^T J_k$ that we use is positive semidefinite. The unconstrained minimizer of $m_k(p)$ is unique when J_k is nonsingular. In this case, we denote it by p_k^{J} and write

$$p_k^{\mathrm{J}} = -(J_k^T J_k)^{-1}(J_k^T r_k) = -J_k^{-1} r_k.$$

The selection of p_k in the dogleg method proceeds as follows.

Procedure 11.6 (Dogleg).
Calculate p_k^{C};
if $\|p_k^{\mathrm{C}}\| = \Delta_k$
    $p_k \leftarrow p_k^{\mathrm{C}}$;
else
    Calculate p_k^{J};
    $p_k \leftarrow p_k^{\mathrm{C}} + \tau(p_k^{\mathrm{J}} - p_k^{\mathrm{C}})$, where τ is the largest value in $[0, 1]$
        such that $\|p_k\| \leq \Delta_k$;
end (if).

Lemma 4.2 shows that when J_k is nonsingular, the vector p_k chosen above is the minimizer of m_k along the piecewise linear path that leads from the origin to the Cauchy point and then to the unconstrained minimizer p_k^{J}. Hence, the reduction in model function at least matches the reduction obtained by the Cauchy point, which can be estimated by specializing the bound (4.20) to the least-squares case by writing

$$m_k(0) - m_k(p_k) \geq c_1 \|J_k^T r_k\| \min\left(\Delta_k, \frac{\|J_k^T r_k\|}{\|J_k^T J_k\|}\right), \tag{11.50}$$

where c_1 is some positive constant.

From Theorem 4.1, we know that the exact solution of (11.46) has the form

$$p_k = -(J_k^T J_k + \lambda_k I)^{-1} J_k^T r_k, \tag{11.51}$$

for some $\lambda_k \geq 0$, and that $\lambda_k = 0$ if the unconstrained solution p_k^{J} satisfies $\|p_k^{\mathrm{J}}\| \leq \Delta_k$. (Note that (11.51) is identical to the formula (10.34a) from Chapter 10. In fact, the Levenberg–Marquardt approach for nonlinear equations is a special case of the same algorithm for nonlinear least-squares problems.) The Levenberg–Marquardt algorithm uses the techniques of Section 4.3 to search for the value of λ_k that satisfies (11.51). The procedure described in the "exact" trust-region algorithm, Algorithm 4.3, is based on Cholesky factorizations, but as in Chapter 10, we can replace these by specialized algorithms to compute the QR factorization of the matrix (11.45). Even if the exact λ_k corresponding to the solution of (11.46) is not found, the p_k calculated from (11.51) will still yield global convergence if it

satisfies the condition (11.50) for some value of c_1, together with

$$\|p_k\| \leq \gamma \Delta_k, \quad \text{for some constant } \gamma \geq 1. \tag{11.52}$$

The dogleg method requires just one linear system to be solved per iteration, whereas methods that search for the exact solution of (11.46) require several such systems to be solved. As in Chapter 4, there is a tradeoff to be made between the amount of effort to spend on each iteration and the total number of function and derivative evaluations required.

We can also consider alternative trust-region approaches that are based on different merit functions and different definitions of the trust region. An algorithm based on the ℓ_1 merit function with an ℓ_∞-norm trust region gives rise to subproblems of the form

$$\min_p \|J_k p + r_k\|_1 \quad \text{subject to } \|p\|_\infty \leq \Delta, \tag{11.53}$$

which can be formulated and solved using linear programming techniques. This approach is closely related to the Sℓ_1QP and SLQP approaches for nonlinear programming discussed in Section 18.5.

Global convergence results of Algorithm 11.5 when the steps p_k satisfy (11.50) and (11.52) are given in the following theorem, which can be proved by referring directly to Theorems 4.5 and 4.6. The first result is for $\eta = 0$, in which the algorithm accepts all steps that produce a decrease in the merit function f_k, while the second (stronger) result requires a strictly positive choice of η.

Theorem 11.7.

Suppose that $J(x)$ is Lipschitz continuous and that $\|J(x)\|$ is bounded above in a neighborhood $\mathcal{D}$ of the level set $\mathcal{L} = \{x : f(x) \leq f(x_0)\}$. Suppose in addition that all approximate solutions of (11.46) satisfy the bounds (11.50) and (11.52). Then if $\eta = 0$ in Algorithm 11.5, we have that

$$\liminf_{k\to\infty} \|J_k^T r_k\| = 0,$$

while if $\eta \in \left(0, \frac{1}{4}\right)$, we have

$$\lim_{k\to\infty} \|J_k^T r_k\| = 0.$$

We turn now to local convergence of the trust-region algorithm for the case in which the subproblem (11.46) is solved exactly. We assume that the sequence $\{x_k\}$ converges to a nondegenerate solution x^* of the nonlinear equations $r(x) = 0$. The significance of this result is that the algorithmic enhancements needed for global convergence do not, in well-designed algorithms, interfere with the fast local convergence properties described in Section 11.1.

Theorem 11.8.

Suppose that the sequence $\{x_k\}$ generated by Algorithm 11.5 converges to a nondegenerate solution x^* of the problem $r(x) = 0$. Suppose also that $J(x)$ is Lipschitz continuous in an open neighborhood $\mathcal{D}$ of x^* and that the trust-region subproblem (11.46) is solved exactly for all sufficiently large k. Then the sequence $\{x_k\}$ converges quadratically to x^*.

PROOF. We prove this result by showing that there is an index K such that the trust-region radius is not reduced further after iteration K; that is, $\Delta_k \geq \Delta_K$ for all $k \geq K$. We then show that the algorithm eventually takes the pure Newton step at every iteration, so that quadratic convergence follows from Theorem 11.2.

Let p_k denote the exact solution of (11.46). Note first that p_k will simply be the unconstrained Newton step $-J_k^{-1} r_k$ whenever this step satisfies the trust-region bound. Otherwise, we have $\|J_k^{-1} r_k\| > \Delta_k$, while the solution p_k satisfies $\|p_k\| \leq \Delta_k$. In either case, we have

$$\|p_k\| \leq \|J_k^{-1} r_k\|. \tag{11.54}$$

We consider the ratio ρ_k of actual to predicted reduction defined by (11.47). We have directly from the definition that

$$|1 - \rho_k| \leq \frac{\left| \|r_k + J_k p_k\|^2 - \|r(x_k + p_k)\|^2 \right|}{\|r(x_k)\|^2 - \|r(x_k) + J(x_k) p_k\|^2}. \tag{11.55}$$

From Theorem 11.1, we have for the second term in the numerator that

$$\|r(x_k + p_k)\|^2 = \|[r(x_k) + J(x_k) p_k] + w(x_k, x_k + p_k)\|^2, \tag{11.56}$$

where $w(\cdot, \cdot)$ is defined as in (11.11). Because of Lipschitz continuity of J with Lipschitz constant β_L (11.7), we have

$$\begin{aligned} \|w(x_k, x_k + p_k)\| &\leq \int_0^1 \|J(x_k + t p_k) - J(x_k)\| \|p_k\| \, dt \\ &\leq \int_0^1 \beta_L \|p_k\|^2 \, dt = (\beta_L/2) \|p_k\|^2, \end{aligned}$$

so that using (11.56) and the fact that $\|r_k + J_k p_k\| \leq \|r_k\| = f(x_k)^{1/2}$ (since p_k is the solution of (11.46)), we can bound the numerator as follows:

$$\begin{aligned} &\left| \|r_k + J_k p_k\|^2 - \|r(x_k + p_k)\|^2 \right| \\ &\leq 2 \|r_k + J_k p_k\| \|w(x_k, x_k + p_k)\| + \|w(x_k, x_k + p_k)\|^2 \\ &\leq f(x_k)^{1/2} \beta_L \|p_k\|^2 + (\beta_L/2)^2 \|p_k\|^4 \\ &\leq \epsilon(x_k) \|p_k\|^2, \end{aligned} \tag{11.57}$$

where we define

$$\epsilon(x_k) = f(x_k)^{1/2}\beta_L + (\beta_L/2)^2\|p_k\|^2.$$

Since $x_k \to x^*$ by assumption, it follows that $f(x_k) \to 0$ and $\|r_k\| \to 0$. Because x^* is a nondegenerate root, we have as in (11.14) that $\|J(x_k)^{-1}\| \leq \beta^*$ for all k sufficiently large, so from (11.54), we have

$$\|p_k\| \leq \|J_k^{-1}r_k\| \leq \beta^*\|r_k\| \to 0. \tag{11.58}$$

Hence, $\epsilon(x_k) \to 0$.

Turning now to the denominator of (11.55), we define $\bar{p}_k$ to be a step of the same length as the solution p_k in the Newton direction $-J_k^{-1}r_k$, that is,

$$\bar{p}_k = -\frac{\|p_k\|}{\|J_k^{-1}r_k\|}J_k^{-1}r_k.$$

Since $\bar{p}_k$ is feasible for (11.46), and since p_k is optimal for this subproblem, we have

$$\begin{aligned}\|r_k\|^2 - \|r_k + J_k p_k\|^2 &\geq \|r_k\|^2 - \left\|r_k - \frac{\|p_k\|}{\|J_k^{-1}r_k\|}r_k\right\|^2 \\ &= 2\frac{\|p_k\|}{\|J_k^{-1}r_k\|}\|r_k\|^2 - \frac{\|p_k\|^2}{\|J_k^{-1}r_k\|^2}\|r_k\|^2 \\ &\geq \frac{\|p_k\|}{\|J_k^{-1}r_k\|}\|r_k\|^2,\end{aligned}$$

where for the last inequality we have used (11.54). By using (11.58) again, we have from this bound that

$$\|r_k\|^2 - \|r_k + J_k p_k\|^2 \geq \frac{\|p_k\|}{\|J_k^{-1}r_k\|}\|r_k\|^2 \geq \frac{1}{\beta^*}\|p_k\|\|r_k\|. \tag{11.59}$$

By substituting (11.57) and (11.59) into (11.55), and then applying (11.58) again, we have

$$|1 - \rho_k| \leq \frac{\beta^*\epsilon(x_k)\|p_k\|^2}{\|p_k\|\|r_k\|} \leq (\beta^*)^2\epsilon(x_k) \to 0. \tag{11.60}$$

Therefore, for all k sufficiently large, we have $\rho_k > \frac{1}{4}$, and so the trust region radius Δ_k will not be increased beyond this point. As claimed, there is an index K such that

$$\Delta_k \geq \Delta_K, \quad \text{for all } k \geq K.$$

Since $\|J_k^{-1} r_k\| \leq \beta^* \|r_k\| \to 0$, the Newton step $-J_k^{-1} r_k$ will eventually be smaller than Δ_K (and hence Δ_k), so it will eventually always be accepted as the solution of (11.46). The result now follows from Theorem 11.2. □

We can replace the assumption that $x_k \to x^*$ with an assumption that the nondegenerate solution x^* is just one of the limit points of the sequence. (In fact, this condition implies that $x_k \to x^*$; see Exercise 11.9.)

11.3 CONTINUATION/HOMOTOPY METHODS

MOTIVATION

We mentioned above that Newton-based methods all suffer from one shortcoming: Unless $J(x)$ is nonsingular in the region of interest—a condition that often cannot be guaranteed—they are in danger of converging to a local minimum of the merit function rather that is not a solution of the nonlinear system. Continuation methods, which we outline in this section, are more likely to converge to a solution of $r(x) = 0$ in difficult cases. Their underlying motivation is simple to describe: Rather than dealing with the original problem $r(x) = 0$ directly, we set up an "easy" system of equations for which the solution is obvious. We then gradually transform the easy system into the original system $r(x)$, and follow the solution as it moves from the solution of the easy problem to the solution of the original problem.

One simple way to define the so-called *homotopy map* $H(x, \lambda)$ is as follows:

$$H(x, \lambda) = \lambda r(x) + (1 - \lambda)(x - a), \tag{11.61}$$

where λ is a scalar parameter and $a \in \mathbb{R}^n$ is a fixed vector. When $\lambda = 0$, (11.61) defines the artificial, easy problem $H(x, 0) = x - a$, whose solution is obviously $x = a$. When $\lambda = 1$, we have $H(x, 1) = r(x)$, the original system of equations.

To solve $r(x) = 0$, consider the following algorithm: First, set $\lambda = 0$ in (11.61) and set $x = a$. Then, increase λ from 0 to 1 in small increments, and for each value of λ, calculate the solution of the system $H(x, \lambda) = 0$. The final value of x corresponding to $\lambda = 1$ will solve the original problem $r(x) = 0$.

This naive approach sounds plausible, and Figure 11.3 illustrates a situation in which it would be successful. In this figure, there is a unique solution x of the system $H(x, \lambda) = 0$ for each value of λ in the range $[0, 1]$. The trajectory of points (x, λ) for which $H(x, \lambda) = 0$ is called the *zero path*.

Unfortunately, however, the approach often fails, as illustrated in Figure 11.4. Here, the algorithm follows the lower branch of the curve from $\lambda = 0$ to $\lambda = \lambda_T$, but it then loses the trail unless it is lucky enough to jump to the top branch of the path. The value λ_T is

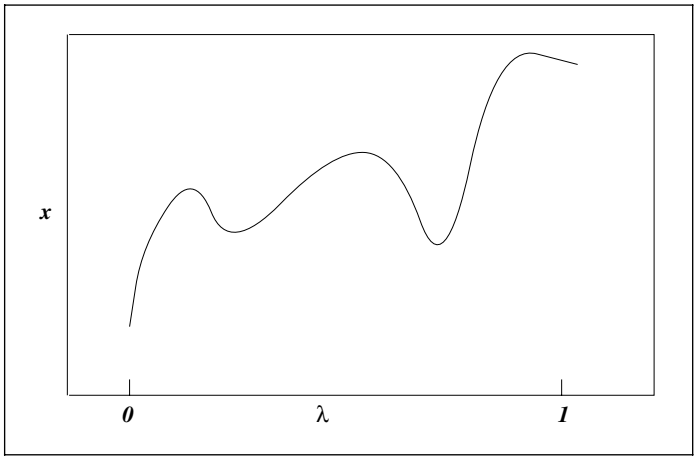

Figure 11.3 Plot of a zero path: Trajectory of points (x, λ) with $H(x, \lambda) = 0$.

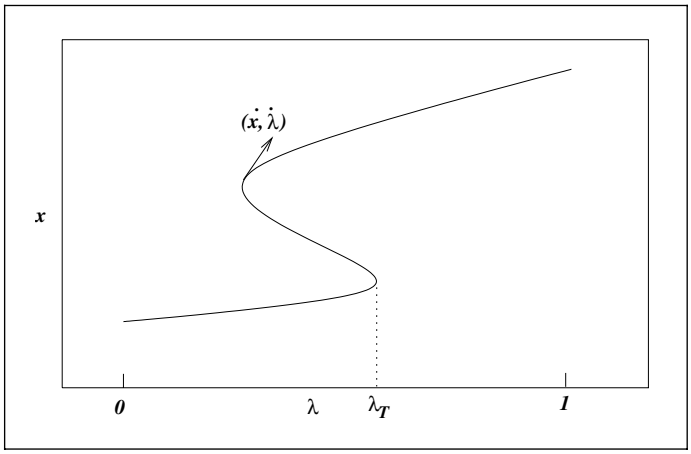

Figure 11.4 Zero path with turning points. The path joining $(a, 0)$ to $(x^*, 1)$ cannot be followed by increasing λ monotonically from 0 to 1.

known as a *turning point*, since at this point we can follow the path smoothly only if we no longer insist on increasing λ at every step. In fact, practical continuation methods work by doing exactly as Figure 11.4 suggests, that is, they follow the zero path explicitly, even if this means allowing λ to decrease from time to time.

PRACTICAL CONTINUATION METHODS

In one practical technique, we model the zero path by allowing both x and λ to be functions of an independent variable s that represents arc length along the path. That is,

$(x(s), \lambda(s))$ is the point that we arrive at by traveling a distance s along the path from the initial point $(x(0), \lambda(0)) = (a, 0)$. Because we have that

$$H(x(s), \lambda(s)) = 0, \quad \text{for all } s \geq 0,$$

we can take the total derivative of this expression with respect to s to obtain

$$\frac{\partial}{\partial x} H(x, \lambda)\dot{x} + \frac{\partial}{\partial \lambda} H(x, \lambda)\dot{\lambda} = 0, \quad \text{where } (\dot{x}, \dot{\lambda}) = \left(\frac{dx}{ds}, \frac{d\lambda}{ds}\right). \tag{11.62}$$

The vector $(\dot{x}(s), \dot{\lambda}(s))$ is the tangent vector to the zero path, as we illustrate in Figure 11.4. From (11.62), we see that it lies in the null space of the $n \times (n+1)$ matrix

$$\begin{bmatrix} \dfrac{\partial}{\partial x} H(x, \lambda) & \dfrac{\partial}{\partial \lambda} H(x, \lambda) \end{bmatrix}. \tag{11.63}$$

When this matrix has full rank, its null space has dimension 1, so to complete the definition of $(\dot{x}, \dot{\lambda})$ in this case, we need to assign it a length and direction. The length is fixed by imposing the normalization condition

$$\|\dot{x}(s)\|^2 + |\dot{\lambda}(s)|^2 = 1, \quad \text{for all } s, \tag{11.64}$$

which ensures that s is the true arc length along the path from $(0, a)$ to $(x(s), \lambda(s))$. We need to choose the sign to ensure that we keep moving forward along the zero path. A heuristic that works well is to choose the sign so that the tangent vector $(\dot{x}, \dot{\lambda})$ at the current value of s makes an angle of less than $\pi/2$ with the tangent point at the previous value of s.

We can outline the complete procedure for computing $(\dot{x}, \dot{\lambda})$ as follows:

Procedure 11.7 (Tangent Vector Calculation).

Compute a vector in the null space of (11.63) by performing a QR factorization with column pivoting,

$$Q^T \begin{bmatrix} \dfrac{\partial}{\partial x} H(x, \lambda) & \dfrac{\partial}{\partial \lambda} H(x, \lambda) \end{bmatrix} \Pi = \begin{bmatrix} R & w \end{bmatrix},$$

where Q is $n \times n$ orthogonal, R is $n \times n$ upper triangular, Π is an $(n+1) \times (n+1)$ permutation matrix, and $w \in \mathbb{R}^n$.

Set

$$v = \Pi \begin{bmatrix} R^{-1}w \\ -1 \end{bmatrix};$$

Set $(\dot{x}, \dot{\lambda}) = \pm v/\|v\|_2$, where the sign is chosen to satisfy the angle criterion mentioned above.

Details of the QR factorization procedure are given in the Appendix.

Since we can obtain the tangent at any given point (x, λ) and since we know the initial point $(x(0), \lambda(0)) = (a, 0)$, we can trace the zero path by calling a standard initial-value first-order ordinary differential equation solver, terminating the algorithm when it finds a value of s for which $\lambda(s) = 1$.

A second approach for following the zero path is quite similar to the one just described, except that it takes an algebraic viewpoint instead of a differential-equations viewpoint. Given a current point (x, λ), we compute the tangent vector $(\dot{x}, \dot{\lambda})$ as above, and take a small step (of length ϵ, say) along this direction to produce a "predictor" point (x^P, λ^P); that is,

$$(x^P, \lambda^P) = (x, \lambda) + \epsilon(\dot{x}, \dot{\lambda}).$$

Usually, this new point will not lie exactly on the zero path, so we apply some "corrector" iterations to bring it back to the path, thereby identifying a new iterate (x^+, λ^+) that satisfies $H(x^+, \lambda^+) = 0$. (This process is illustrated in Figure 11.5.) During the corrections, we choose a component of the predictor step (x^P, λ^P)—one of the components that has been changing most rapidly during the past few steps—and hold this component fixed during the correction process. If the index of this component is i, and if we use a pure Newton corrector process (often adequate, since (x^P, λ^P) is usually quite close to the target point

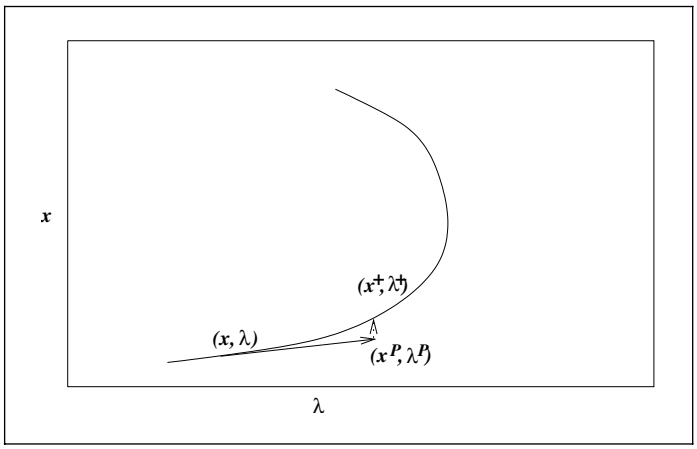

Figure 11.5 The algebraic predictor–corrector procedure, using λ as the fixed variable in the correction process.

(x^+, λ^+)), the steps will have the form

$$\left[\begin{array}{c|c} \dfrac{\partial H}{\partial x} & \dfrac{\partial H}{\partial \lambda} \\ \hline \multicolumn{2}{c}{e_i} \end{array}\right] \left[\begin{array}{c} \delta x \\ \delta \lambda \end{array}\right] = \left[\begin{array}{c} -H \\ 0 \end{array}\right],$$

where the quantities $\partial H/\partial x$, $\partial H/\partial \lambda$, and H are evaluated at the latest point of the corrector process. The last row of this system serves to fix the ith component of $(\delta x, \delta \lambda)$ at zero; the vector $e_i \in \mathbb{R}^{n+1}$ is a vector with $n+1$ components containing all zeros, except for a 1 in the location i that corresponds to the fixed component. Note that in Figure 11.5 the λ component is chosen to be fixed on the current iteration. On the following iteration, it may be more appropriate to choose x as the fixed component, as we reach the turning point in λ.

The two variants on path-following described above are able to follow curves like those depicted in Figure 11.4 to a solution of the nonlinear system. They rely, however, on the $n \times (n+1)$ matrix in (11.63) having full rank for all (x, λ) along the path, so that the tangent vector is well-defined. The following result shows that full rank is guaranteed under certain assumptions.

Theorem 11.9 (Watson [305]).

Suppose that r is twice continuously differentiable. Then for almost all vectors $a \in \mathbb{R}^n$, there is a zero path emanating from $(0, a)$ along which the $n \times (n+1)$ matrix (11.63) has full rank. If this path is bounded for $\lambda \in [0, 1)$, then it has an accumulation point $(\bar{x}, 1)$ such that $r(\bar{x}) = 0$. Furthermore, if the Jacobian $J(\bar{x})$ is nonsingular, the zero path between $(a, 0)$ and $(\bar{x}, 1)$ has finite arc length.

The theorem assures us that unless we are unfortunate in the choice of a, the algorithms described above can be applied to obtain a path that either diverges or else leads to a point $\bar{x}$ that is a solution of the original nonlinear system if $J(\bar{x})$ is nonsingular. More detailed convergence results can be found in Watson [305] and the references therein.

We conclude with an example to show that divergence of the zero path—the less desirable outcome of Theorem 11.9—can happen even for innocent-looking problems.

❑ EXAMPLE 11.3

Consider the system $r(x) = x^2 - 1$, for which there are two nondegenerate solutions $+1$ and -1. Suppose we choose $a = -2$ and attempt to apply a continuation method to the function

$$H(x, \lambda) = \lambda(x^2 - 1) + (1 - \lambda)(x + 2) = \lambda x^2 + (1 - \lambda)x + (2 - 3\lambda), \qquad (11.65)$$

obtained by substituting into (11.61). The zero paths for this function are plotted in Figure 11.6. As can be seen from that diagram, there is no zero path that joins $(-2, 0)$

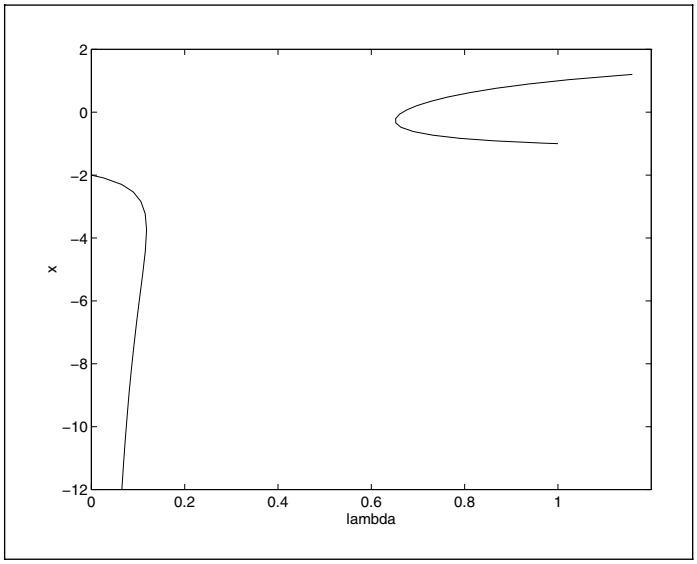

Figure 11.6 Zero paths for the example in which $H(x, \lambda) = \lambda(x^2-1)+(1-\lambda)(x+2)$. There is no continuous zero path from $\lambda = 0$ to $\lambda = 1$.

to either $(1, 1)$ or $(-1, 1)$, so the continuation methods fail on this example. We can find the values of λ for which no solution exists by using the formula for a quadratic root to obtain

$$x = \frac{-(1-\lambda) \pm \sqrt{(1-\lambda)^2 - 4\lambda(2-3\lambda)}}{2\lambda}.$$

Now, when the term in the square root is negative, the corresponding values of x are complex, that is, there are no real roots x. It is easy to verify that such is the case when

$$\lambda \in \left(\frac{5-2\sqrt{3}}{13}, \frac{5+2\sqrt{3}}{13} \right) \approx (0.118, 0.651).$$

Note that the zero path starting from $(-2, 0)$ becomes unbounded, which is one of the possible outcomes of Theorem 11.9.

❐

This example indicates that continuation methods may fail to produce a solution even to a fairly simple system of nonlinear equations. However, it is generally true that they are more reliable than the merit-function methods described earlier in the chapter. The extra robustness comes at a price, since continuation methods typically require significantly more computational effort than the merit-function methods.

NOTES AND REFERENCES

Nonlinear differential equations and integral equations are a rich source of nonlinear equations. When formulated as finite-dimensional nonlinear equations, the unknown vector x is a discrete approximation to the (infinite-dimensional) solution. In other applications, the vector x is intrinsically finite-dimensional; it may represent the quantities of materials to be transported between pairs of cities in a distribution network, for instance. In all cases, the equations r_i enforce consistency, conservation, and optimality principles in the model. Moré [212] and Averick et al. [10] discuss a number of interesting practical applications.

For analysis of the convergence of Broyden's method, including proofs of Theorem 11.5, see Dennis and Schnabel [92, Chapter 8] and Kelley [177, Chapter 6]. Details on a limited-memory implementation of Broyden's method are given by Kelley [177, Section 7.3].

Example 11.2 and the algorithm described by Powell [241] have been influential beyond the field of nonlinear equations. The example shows that a line-search method may not be able to achieve sufficient decrease, whereas the Cauchy step in the trust-region approach is designed to guarantee that this condition holds and hence that reasonable convergence properties are guaranteed. The dogleg algorithm proposed in [241] can be viewed as one of the first modern trust-region methods.

EXERCISES

11.1 Show that for any vector $s \in \mathbb{R}^n$, we have

$$\left\| \frac{ss^T}{s^T s} \right\| = 1,$$

where $\|\cdot\|$ denotes the Euclidean matrix norm.

11.2 Consider the function $r : \mathbb{R} \to \mathbb{R}$ defined by $r(x) = x^q$, where q is an integer greater than 2. Note that $x^* = 0$ is the sole root of this function and that it is degenerate. Show that Newton's method converges Q-linearly, and find the value of the convergence ratio r in (A.34).

11.3 Show that Newton's method applied to the function $r(x) = -x^5 + x^3 + 4x$ starting from $x_0 = 1$ produces the cyclic behavior described in the text. Find the roots of this function, and check that they are nondegenerate.

11.4 For the scalar function $r(x) = \sin(5x) - x$, show that the sum-of-squares merit function has infinitely many local minima, and find a general formula for such points.

11.5 When $r : \mathbb{R}^n \to \mathbb{R}^n$, show that the function

$$\phi(\lambda) = \left\| (J^T J + \lambda I)^{-1} J^T r \right\|$$

is monotonically decreasing in λ unless $J^T r = 0$. (Hint: Use the singular-value decomposition of J.)

✎ **11.6** Prove part (iii) of Theorem 11.3.

✎ **11.7** Consider a line-search Newton method in which the step length α_k is chosen to be the exact minimizer of the merit function $f(\cdot)$; that is,

$$\alpha_k = \arg\min_{\alpha} f(x_k - \alpha J_k^{-1} r_k).$$

Show that if $J(x)$ is nonsingular at the solution x^*, then $\alpha_k \to 1$ as $x_k \to x^*$.

✎ **11.8** Let $J \in \mathbb{R}^{n\times m}$ and $r \in \mathbb{R}^n$ and suppose that $JJ^T r = 0$. Show that $J^T r = 0$. (Hint: This doesn't even take one line!)

✎ **11.9** Suppose we replace the assumption of $x_k \to x^*$ in Theorem 11.8 by an assumption that the nondegenerate solution x^* is a limit point of x^*. By adding some logic to the proof of this result, show that in fact x^* is the only possible limit point of the sequence. (Hint: Show that $\|J_{k+1}^{-1} r_{k+1}\| \leq \frac{1}{2}\|J_k^{-1} r_k\|$ for all k sufficiently large, and hence that for any constant $\epsilon > 0$, the sequence $\{x_k\}$ satisfies $\|x_k - x^*\| \leq \epsilon$ for all k sufficiently large.)

✎ **11.10** Consider the following modification of our example of failure of continuation methods:

$$r(x) = x^2 - 1, \quad a = \tfrac{1}{2}.$$

Show that for this example there is a zero path for $H(x, \lambda) = \lambda(x^2 - 1) + (1 - \lambda)(x - a)$ that connects $(\frac{1}{2}, 0)$ to $(1, 1)$, so that continuation methods should work for this choice of starting point.