APPENDIX A

Background Material

A.1 ELEMENTS OF LINEAR ALGEBRA

VECTORS AND MATRICES

In this book we work exclusively with vectors and matrices whose components are real numbers. Vectors are usually denoted by lowercase roman characters, and matrices by uppercase roman characters. The space of real vectors of length n is denoted by $\mathbb{R}^n$, while the space of real $m \times n$ matrices is denoted by $\mathbb{R}^{m \times n}$.

Given a vector $x \in \mathbb{R}^n$, we use x_i to denote its ith component. We invariably assume that x is a *column* vector, that is,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

The transpose of x, denoted by x^T is the row vector

$$x^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix},$$

and is often also written with parentheses as $x = (x_1, x_2, \ldots, x_n)$. We write $x \geq 0$ to indicate componentwise nonnegativity, that is, $x_i \geq 0$ for all $i = 1, 2, \ldots, n$, while $x > 0$ indicates that $x_i > 0$ for all $i = 1, 2, \ldots, n$.

Given $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, the standard inner product is $x^T y = \sum_{i=1}^n x_i y_i$.

Given a matrix $A \in \mathbb{R}^{m \times n}$, we specify its components by double subscripts as A_{ij}, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. The transpose of A, denoted by A^T, is the $n \times m$ matrix whose components are A_{ji}. The matrix A is said to be *square* if $m = n$. A square matrix is *symmetric* if $A = A^T$.

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

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

It is *positive semidefinite* if

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

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.

The diagonal of the matrix $A \in \mathbb{R}^{m \times n}$ consists of the elements A_{ii}, for $i = 1, 2, \ldots \min(m, n)$. The matrix $A \in \mathbb{R}^{m \times n}$ is *lower triangular* if $A_{ij} = 0$ whenever $i < j$; that is, all elements above the diagonal are zero. It is upper triangular if $A_{ij} = 0$ whenever $i > j$; that is, all elements below the diagonal are zero. A is *diagonal* if $A_{ij} = 0$ whenever $i \neq j$.

The identity matrix, denoted by I, is the square diagonal matrix whose diagonal elements are all 1.

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.

NORMS

For a vector $x \in \mathbb{R}^n$, we define the following norms:

$$\|x\|_1 \stackrel{\text{def}}{=} \sum_{i=1}^{n} |x_i|, \tag{A.2a}$$

$$\|x\|_2 \stackrel{\text{def}}{=} \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2} = (x^T x)^{1/2}, \tag{A.2b}$$

$$\|x\|_\infty \stackrel{\text{def}}{=} \max_{i=1,\ldots,n} |x_i|. \tag{A.2c}$$

The norm $\|\cdot\|_2$ is often called the *Euclidean norm*. We sometimes refer to $\|\cdot\|_1$ as the ℓ_1 norm and to $\|\cdot\|_\infty$ as the ℓ_∞ norm. All these norms measure the length of the vector in some sense, and they are equivalent in the sense that each one is bounded above and below by a multiple of the other. To be precise, we have for all $x \in \mathbb{R}^n$ that

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty, \qquad \|x\|_\infty \leq \|x\|_1 \leq n\|x\|_\infty, \tag{A.3}$$

and so on. In general, a norm is any mapping $\|\cdot\|$ from $\mathbb{R}^n$ to the nonnegative real numbers that satisfies the following properties:

$$\|x + z\| \leq \|x\| + \|z\|, \qquad \text{for all } x, z \in \mathbb{R}^n; \tag{A.4a}$$

$$\|x\| = 0 \Rightarrow x = 0; \tag{A.4b}$$

$$\|\alpha x\| = |\alpha|\|x\|, \qquad \text{for all } \alpha \in \mathbb{R} \text{ and } x \in \mathbb{R}^n. \tag{A.4c}$$

Equality holds in (A.4a) if and only if one of the vectors x and z is a nonnegative scalar multiple of the other.

Another interesting property that holds for the Euclidean norm $\|\cdot\| = \|\cdot\|_2$ is the Cauchy–Schwarz inequality, which states that

$$\left|x^T z\right| \leq \|x\|\,\|z\|, \tag{A.5}$$

with equality if and only if one of these vectors is a nonnegative multiple of the other. We can prove this result as follows:

$$0 \leq \|\alpha x + z\|^2 = \alpha^2\|x\|^2 + 2\alpha x^T z + \|z\|^2.$$

The right-hand-side is a convex function of α, and it satisfies the required nonnegativity property only if there exist fewer than 2 distinct real roots, that is,

$$(2x^T z)^2 \leq 4\|x\|^2\|z\|^2,$$

proving (A.5). Equality occurs when the quadratic α has exactly one real root (that is, $|x^T z| = \|x\| \, \|z\|$) and when $\alpha x + z = 0$ for some α, as claimed.

Any norm $\| \cdot \|$ has a *dual norm* $\| \cdot \|_D$ defined by

$$\|x\|_D = \max_{\|y\|=1} x^T y. \tag{A.6}$$

It is easy to show that the norms $\| \cdot \|_1$ and $\| \cdot \|_\infty$ are duals of each other, and that the Euclidean norm is its own dual.

We can derive definitions for certain matrix norms from these vector norm definitions. If we let $\| \cdot \|$ be generic notation for the three norms listed in (A.2), we define the corresponding matrix norm as

$$\|A\| \stackrel{\text{def}}{=} \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|}. \tag{A.7}$$

The matrix norms defined in this way are said to be *consistent* with the vector norms (A.2). Explicit formulae for these norms are as follows:

$$\|A\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{m} |A_{ij}|, \tag{A.8a}$$

$$\|A\|_2 = \text{largest eigenvalue of } (A^T A)^{1/2}, \tag{A.8b}$$

$$\|A\|_\infty = \max_{i=1,\ldots,m} \sum_{j=1}^{n} |A_{ij}|. \tag{A.8c}$$

The Frobenius norm $\|A\|_F$ of the matrix A is defined by

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}^2 \right)^{1/2}. \tag{A.9}$$

This norm is useful for many purposes, but it is not consistent with any vector norm. Once again, these various matrix norms are equivalent with each other in a sense similar to (A.3).

For the Euclidean norm $\| \cdot \| = \| \cdot \|_2$, the following property holds:

$$\|AB\| \leq \|A\| \, \|B\|, \tag{A.10}$$

for all matrices A and B with consistent dimensions.

The *condition number* of a nonsingular matrix is defined as

$$\kappa(A) = \|A\| \, \|A^{-1}\|, \tag{A.11}$$

where any matrix norm can be used in the definition. Different norms can by the use of a subscript—$\kappa_1(\cdot)$, $\kappa_2(\cdot)$, and $\kappa_\infty(\cdot)$, respectively—with κ denoting κ_2 by default.

Norms also have a meaning for scalar, vector, and matrix-valued functions that are defined on a particular domain. In these cases, we can define Hilbert spaces of functions for which the inner product and norm are defined in terms of an integral over the domain. We omit details, since all the development of this book takes place in the space $\mathbb{R}^n$, though many of the algorithms can be extended to more general Hilbert spaces. However, we mention for purposes of the analysis of Newton-like methods that the following inequality holds for functions of the type that we consider in this book:

$$\left\| \int_a^b F(t) \right\| \leq \int_a^b \|F(t)\| \, dt, \tag{A.12}$$

where F is a continuous scalar-, vector-, or matrix-valued function on the interval $[a, b]$.

SUBSPACES

Given the Euclidean space $\mathbb{R}^n$, the subset $\mathcal{S} \subset \mathbb{R}^n$ is a *subspace of* $\mathbb{R}^n$ if the following property holds: If x and y are any two elements of $\mathcal{S}$, then

$$\alpha x + \beta y \in \mathcal{S}, \quad \text{for all } \alpha, \beta \in \mathbb{R}.$$

For instance, $\mathcal{S}$ is a subspace of $\mathbb{R}^2$ if it consists of (i) the whole space $\mathbb{R}^n$; (ii) any line passing through the origin; (iii) the origin alone; or (iv) the empty set.

Given any set of vectors $a_i \in \mathbb{R}^n$, $i = 1, 2, \ldots, m$, the set

$$\mathcal{S} = \left\{ w \in \mathbb{R}^n \mid a_i^T w = 0, \ i = 1, 2, \ldots, m \right\} \tag{A.13}$$

is a subspace. However, the set

$$\left\{ w \in \mathbb{R}^n \mid a_i^T w \geq 0, \ i = 1, 2, \ldots, m \right\} \tag{A.14}$$

is not in general a subspace. For example, if we have $n = 2, m = 1$, and $a_1 = (1, 0)^T$, this set would consist of all vectors $(w_1, w_2)^T$ with $w_1 \geq 0$, but then given two vectors $x = (1, 0)^T$ and $y = (2, 3)$ in this set, it is easy to choose multiples α and β such that $\alpha x + \beta y$ has a negative first component, and so lies outside the set.

Sets of the forms (A.13) and (A.14) arise in the discussion of second-order optimality conditions for constrained optimization.

A set of vectors $\{s_1, s_2, \ldots, s_m\}$ in $\mathbb{R}^n$ is called a *linearly independent set* if there are no real numbers $\alpha_1, \alpha_2, \ldots, \alpha_m$ such that

$$\alpha_1 s_2 + \alpha_2 s_2 + \cdots + \alpha_m s_m = 0,$$

unless we make the trivial choice $\alpha_1 = \alpha_2 = \cdots = \alpha_m = 0$. Another way to define linear independence is to say that none of the vectors $s_1, s_2, \ldots, s_m$ can be written as a linear combination of the other vectors in this set. If in fact we have $s_i \in \mathcal{S}$ for all $i = 1, 2, \ldots, m$, we say that $\{s_1, s_2, \ldots, s_m\}$ is a *spanning set* for $\mathcal{S}$ if *any* vector $s \in \mathcal{S}$ can be written as

$$s = \alpha_1 s_2 + \alpha_2 s_2 + \cdots + \alpha_m s_m,$$

for some particular choice of the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_m$.

If the vectors $s_1, s_2, \ldots, s_m$ are both linearly independent and a spanning set for $\mathcal{S}$, we call them a *basis* of $\mathcal{S}$. In this case, m (the number of elements in the basis) is referred to as the *dimension* of $\mathcal{S}$, and denoted by $\dim(\mathcal{S})$. Note that there are many ways to choose a basis of $\mathcal{S}$ in general, but that all bases contain the same number of vectors.

If A is any real matrix, the *null space* is the subspace

$$\text{Null}(A) = \{w \mid Aw = 0\},$$

while the *range space* is

$$\text{Range}(A) = \{w \mid w = Av \text{ for some vector } v\}.$$

The *fundamental theorem of linear algebra* states that

$$\text{Null}(A) \oplus \text{Range}(A^T) = \mathbb{R}^n,$$

where n is the number of columns in A. (Here, "$\oplus$" denotes the direct sum of two sets: $\mathcal{A} \oplus \mathcal{B} = \{x + y \mid x \in \mathcal{A}, y \in \mathcal{B}\}$.)

When A is square ($n \times n$) and nonsingular, we have $\text{Null}A = \text{Null}A^T = \{0\}$ and $\text{Range}A = \text{Range}A^T = \mathbb{R}^n$. In this case, the columns of A form a basis of $\mathbb{R}^n$, as do the columns of A^T.

EIGENVALUES, EIGENVECTORS, AND THE SINGULAR-VALUE DECOMPOSITION

A scalar value λ is an *eigenvalue* of the $n \times n$ matrix A if there is a nonzero vector q such that

$$Aq = \lambda q.$$

The vector q is called an *eigenvector* of A. The matrix A is nonsingular if none of its eigenvalues are zero. The eigenvalues of symmetric matrices are all real numbers, while nonsymmetric matrices may have imaginary eigenvalues. If the matrix is positive definite as well as symmetric, its eigenvalues are all *positive* real numbers.

All matrices A (not necessarily square) can be decomposed as a product of three matrices with special properties. When $A \in \mathbb{R}^{m \times n}$ with $m > n$, (that is, A has more rows than columns), this *singular-value decomposition (SVD)* has the form

$$A = U \begin{bmatrix} S \\ 0 \end{bmatrix} V^T, \tag{A.15}$$

where U and V are orthogonal matrices of dimension $m \times m$ and $n \times n$, respectively, and S is an $n \times n$ diagonal matrix with diagonal elements σ_i, $i = 1, 2, \ldots, n$, that satisfy

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0.$$

These diagonal values are called the singular values of A. We can define the condition number (A.11) of the $m \times n$ (possibly nonsquare) matrix A to be σ_1/σ_n. (This definition is identical to $\kappa_2(A)$ when A happens to be square and nonsingular.)

When $m \leq n$ (the number of columns is at least equal to the number of rows), the SVD has the form

$$A = U \begin{bmatrix} S & 0 \end{bmatrix} V^T,$$

where again U and V are orthogonal of dimension $m \times m$ and $n \times n$, respectively, while S is $m \times m$ diagonal with nonnegative diagonal elements $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m$.

When A is symmetric, its n real eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and their associated eigenvectors $q_1, q_2, \ldots, q_n$ can be used to write a *spectral decomposition* of A as follows:

$$A = \sum_{i=1}^{n} \lambda_i q_i q_i^T.$$

This decomposition can be restated in matrix form by defining

$$\Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n), \qquad Q = [q_1 \,|\, q_2 \,|\, \ldots \,|\, q_n],$$

and writing

$$A = Q \Lambda Q^T. \tag{A.16}$$

In fact, when A is positive definite as well as symmetric, this decomposition is identical to the singular-value decomposition (A.15), where we define $U = V = Q$ and $S = \Lambda$. Note that the singular values σ_i and the eigenvalues λ_i coincide in this case.

In the case of the Euclidean norm (A.8b), we have for symmetric positive definite matrices A that the singular values and eigenvalues of A coincide, and that

$$\|A\| = \sigma_1(A) = \text{largest eigenvalue of } A,$$
$$\|A^{-1}\| = \sigma_n(A)^{-1} = \text{inverse of smallest eigenvalue of } A.$$

Hence, we have for all $x \in \mathbb{R}^n$ that

$$\sigma_n(A)\|x\|^2 = \|x\|^2/\|A^{-1}\| \leq x^T Ax \leq \|A\|\|x\|^2 = \sigma_1(A)\|x\|^2.$$

For an orthogonal matrix Q, we have for the Euclidean norm that

$$\|Qx\| = \|x\|,$$

and that all the singular values of this matrix are equal to 1.

DETERMINANT AND TRACE

The trace of an $n \times n$ matrix A is defined by

$$\text{trace}(A) = \sum_{i=1}^{n} A_{ii}. \tag{A.17}$$

If the eigenvalues of A are denoted by $\lambda_1, \lambda_2, \ldots, \lambda_n$, it can be shown that

$$\text{trace}(A) = \sum_{i=1}^{n} \lambda_i, \tag{A.18}$$

that is, the trace of the matrix is the sum of its eigenvalues.

The determinant of an $n \times n$ matrix A, denoted by $\det A$, is the product of its eigenvalues; that is,

$$\det A = \prod_{i=1}^{n} \lambda_i. \tag{A.19}$$

The determinant has several appealing (and revealing) properties. For instance,

$\det A = 0$ if and only if A is singular;

$\det AB = (\det A)(\det B)$;

$\det A^{-1} = 1/\det A$.

Recall that any orthogonal matrix A has the property that $QQ^T = Q^TQ = I$, so that $Q^{-1} = Q^T$. It follows from the property of the determinant that $\det Q = \det Q^T = \pm 1$.

The properties above are used in the analysis of Chapter 6.

MATRIX FACTORIZATIONS: CHOLESKY, LU, QR

Matrix factorizations are important both in the design of algorithms and in their analysis. One such factorization is the singular-value decomposition defined above in (A.15). Here we define the other important factorizations.

All the factorization algorithms described below make use of *permutation matrices*. Suppose that we wish to exchange the first and fourth rows of a matrix A. We can perform this operation by premultiplying A by a permutation matrix P, which is constructed by interchanging the first and fourth rows of an identity matrix that contains the same number of rows as A. Suppose, for example, that A is a 5×5 matrix. The appropriate choice of P would be

$$P = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

A similar technique is used to to find a permutation matrix P that exchanges columns of a matrix.

The LU factorization of a matrix $A \in \mathbb{R}^{n \times n}$ is defined as

$$PA = LU, \tag{A.20}$$

where

P is an $n \times n$ permutation matrix (that is, it is obtained by rearranging the rows of the $n \times n$ identity matrix),

L is unit lower triangular (that is, lower triangular with diagonal elements equal to 1, and

U is upper triangular.

This factorization can be used to solve a linear system of the form $Ax = b$ efficiently by the following three-step process:

form $\tilde{b} = Pb$ by permuting the elements of b;

solve $Lz = \tilde{b}$ by performing triangular forward-substitution, to obtain the vector z;

solve $Ux = z$ by performing triangular back-substitution, to obtain the solution vector x.

The factorization (A.20) can be found by using Gaussian elimination with row partial pivoting, an algorithm that requires approximately $2n^3/3$ floating-point operations when A is dense. Standard software that implements this algorithm (notably, LAPACK [7]) is readily available. The method can be stated as follows.

Algorithm A.1 (Gaussian Elimination with Row Partial Pivoting).

```
Given A ∈ R^{n×n};
Set P ← I, L ← 0;
for i = 1, 2, ..., n
    find the index j ∈ {i, i+1, ..., n} such that |A_ji| = max_{k=i,i+1,...,n} |A_ki|;
    if A_ij = 0
        stop; (* matrix A is singular *)
    if i ≠ j
        swap rows i and j of matrices A and L;
    (* elimination step *)
    L_ii ← 1;
    for k = i + 1, i + 2, ..., n
        L_ki ← A_ki / A_ii;
        for l = i + 1, i + 2, ..., n
            A_kl ← A_kl − L_ki A_il;
        end (for)
    end (if)
end (for)
U ← upper triangular part of A.
```

Variants of the basic algorithm allow for rearrangement of the columns as well as the rows during the factorization, but these do not add to the practical stability properties of the algorithm. Column pivoting may, however, improve the performance of Gaussian elimination when the matrix A is sparse. by ensuring that the factors L and U are also reasonably sparse.

Gaussian elimination can be applied also to the case in which A is not square. When A is $m \times n$, with $m > n$, the standard row pivoting algorithm produces a factorization of the form (A.20), where $L \in \mathbb{R}^{m\times n}$ is unit lower triangular and $U \in \mathbb{R}^{n\times n}$ is upper triangular. When $m < n$, we can find an LU factorization of A^T rather than A, that is, we obtain

$$PA^T = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} U, \tag{A.21}$$

where L_1 is $m \times m$ (square) unit lower triangular, U is $m \times m$ upper triangular, and L_2 is a general $(n-m) \times m$ matrix. If A has full row rank, we can use this factorization to calculate

its null space explicitly as the space spanned by the columns of the matrix

$$M = P^T \begin{bmatrix} L_1^{-T} L_2^T \\ -I \end{bmatrix} U^{-T}. \tag{A.22}$$

It is easy to check that M has dimensions $n \times (n-m)$ and that $AM = 0$.

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.) The algorithm can be specified as follows.

Algorithm A.2 (Cholesky Factorization).
Given $A \in \mathbb{R}^{n \times n}$ symmetric positive definite;
for $i = 1, 2, \ldots, n$;
    $L_{ii} \leftarrow \sqrt{A_{ii}}$;
    for $j = i+1, i+2, \ldots, n$
        $L_{ji} \leftarrow A_{ji}/L_{ii}$;
        for $k = i+1, i+2, \ldots, j$
            $A_{jk} \leftarrow A_{jk} - L_{ji}L_{ki}$;
        end (for)
    end (for)
end (for)

Note that this algorithm references only the lower triangular elements of A; in fact, it is only necessary to store these elements in any case, since by symmetry they are simply duplicated in the upper triangular positions.

Unlike the case of Gaussian elimination, the Cholesky algorithm can produce a valid factorization of a symmetric positive definite matrix without swapping any rows or columns. However, symmetric permutation (that is, reordering the rows and columns in the same way) can be used to improve the sparsity of the factor L. In this case, the algorithm produces a permutation of the form

$$P^T A P = LL^T$$

for some permutation matrix P.

The Cholesky factorization can be used to compute solutions of the system $Ax = b$ by performing triangular forward- and back-substitutions with L and L^T, respectively, as in the case of L and U factors produced by Gaussian elimination.

The Cholesky factorization can also be used to verify positive definiteness of a symmetric matrix A. If Algorithm A.2 runs to completion with all L_{ii} values well defined and positive, then A is positive definite.

Another useful factorization of rectangular matrices $A \in \mathbb{R}^{m\times n}$ has the form

$$AP = QR, \tag{A.24}$$

where

P is an $n \times n$ permutation matrix,

A is $m \times m$ orthogonal, and

R is $m \times n$ upper triangular.

In the case of a square matrix $m = n$, this factorization can be used to compute solutions of linear systems of the form $Ax = b$ via the following procedure:

set $\tilde{b} = Q^T b$;

solve $Rz = \tilde{b}$ for z by performing back-substitution;

set $x = P^T z$ by rearranging the elements of x.

For a dense matrix A, the cost of computing the QR factorization is about $4m^2n/3$ operations. In the case of a square matrix, the operation count is about twice as high as for an LU factorization via Gaussian elimination. Moreover, it is more difficult to maintain sparsity in a QR factorization than in an LU factorization.

Algorithms to perform QR factorization are almost as simple as algorithms for Gaussian elimination and for Cholesky factorization. The most widely used algorithms work by applying a sequence of special orthogonal matrices to A, known either as Householder transformations or Givens rotations, depending on the algorithm. We omit the details, and refer instead to Golub and Van Loan [136, Chapter 5] for a complete description.

In the case of a rectangular matrix A with $m < n$, we can use the QR factorization of A^T to find a matrix whose columns span the null space of A. To be specific, we write

$$A^T P = QR = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} R,$$

where Q_1 consists of the first m columns of Q, and Q_2 contains the last $n - m$ columns. It is easy to show that columns of the matrix Q_2 span the null space of A. This procedure yields a more satisfactory basis matrix for the null space than the Gaussian elimination procedure (A.22), because the columns of Q_2 are orthogonal to each other and have unit length. It may be more expensive to compute, however, particularly in the case in which A is sparse.

When A has full column rank, we can make an identification between the R factor in (A.24) and the Cholesky factorization. By multiplying the formula (A.24) by its transpose,

we obtain

$$P^T A^T A P = R^T Q^T Q R = R^T R,$$

and by comparison with (A.23), we see that R^T is simply the Cholesky factor of the symmetric positive definite matrix $P^T A^T A P$. Recalling that L is uniquely defined when we restrict its diagonal elements to be positive, this observation implies that R is also uniquely defined for a given choice of permutation matrix P, provided that we enforce positiveness of the diagonals of R. Note, too, that since we can rearrange (A.24) to read $APR^{-1} = Q$, we can conclude that Q is also uniquely defined under these conditions.

Note that by definition of the Euclidean norm and the property (A.10), and the fact that the Euclidean norms of the matrices P and Q in (A.24) are both 1, we have that

$$\|A\| = \|QRP^T\| \leq \|Q\| \, \|R\| \, \|P^T\| = \|R\|,$$

while

$$\|R\| = \|Q^T A P\| \leq \|Q^T\| \, \|A\| \, \|P\| = \|A\|.$$

We conclude from these two inequalities that $\|A\| = \|R\|$. When A is square, we have by a similar argument that $\|A^{-1}\| = \|R^{-1}\|$. Hence the Euclidean-norm condition number of A can be estimated by substituting R for A in the expression (A.11). This observation is significant because various techniques are available for estimating the condition number of triangular matrices R; see Golub and Van Loan [136, pp. 128–130] for a discussion.

SYMMETRIC INDEFINITE FACTORIZATION

When matrix A is symmetric but indefinite, Algorithm A.2 will break down by trying to take the square root of a negative number. We can however produce a factorization, similar to the Cholesky factorization, of the form

$$PAP^T = LBL^T, \tag{A.25}$$

where L is unit lower triangular, B is a block diagonal matrix with blocks of dimension 1 or 2, and P is a permutation matrix. The first step of this symmetric indefinite factorization proceeds as follows. We identify a submatrix E of A that is suitable to be used as a pivot block. The precise criteria that can be used to choose E are described below, but we note here that E is either a single diagonal element of A (a 1×1 pivot block), or else the 2×2 block consisting of two diagonal elements of A (say, a_{ii} and a_{jj}) along with the corresponding off-diagonal elements (that is, a_{ij} and a_{ji}). In either case, E must be nonsingular. We then

find a permutation matrix P_1 that makes E a leading principal submatrix of A, that is,

$$P_1 A P_1 = \begin{bmatrix} E & C^T \\ C & H \end{bmatrix}, \tag{A.26}$$

and then perform a block factorization on this rearranged matrix, using E as the pivot block, to obtain

$$P_1 A P_1^T = \begin{bmatrix} I & 0 \\ CE^{-1} & I \end{bmatrix} \begin{bmatrix} E & 0 \\ 0 & H - CE^{-1}C^T \end{bmatrix} \begin{bmatrix} I & E^{-1}C^T \\ 0 & I \end{bmatrix}.$$

The next step of the factorization consists in applying exactly the same process to $H - CE^{-1}C^T$, known as the *remaining matrix* or the *Schur complement*, which has dimension either $(n-1) \times (n-1)$ or $(n-2) \times (n-2)$. We now apply the same procedure recursively, terminating with the factorization (A.25). Here P is defined as a product of the permutation matrices from each step of the factorization, and B contains the pivot blocks E on its diagonal.

The symmetric indefinite factorization requires approximately $n^3/3$ floating-point operations—the same as the cost of the Cholesky factorization of a positive definite matrix—but to this count we must add the cost of identifying suitable pivot blocks E and of performing the permutations, which can be considerable. There are various strategies for determining the pivot blocks, which have an important effect on both the cost of the factorization and its numerical properties. Ideally, our strategy for choosing E at each step of the factorization procedure should be inexpensive, should lead to at most modest growth in the elements of the remaining matrix at each step of the factorization, and should avoid excessive fill-in (that is, L should not be too much more dense than A).

A well-known strategy, due to Bunch and Parlett [43], searches the whole remaining matrix and identifies the largest-magnitude diagonal and largest-magnitude off-diagonal elements, denoting their respective magnitudes by ξ_{dia} and ξ_{off}. If the diagonal element whose magnitude is ξ_{dia} is selected to be a 1×1 pivot block, the element growth in the remaining matrix is bounded by the ratio $\xi_{\text{dia}}/\xi_{\text{off}}$. If this growth rate is acceptable, we choose this diagonal element to be the pivot block. Otherwise, we select the off-diagonal element whose magnitude is ξ_{off} (a_{ij}, say), and choose E to be the 2×2 submatrix that includes this element, that is,

$$E = \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ij} & a_{jj} \end{bmatrix}.$$

This pivoting strategy of Bunch and Parlett is numerically stable and guarantees to yield a matrix L whose maximum element is bounded by 2.781. Its drawback is that the evaluation of ξ_{dia} and ξ_{off} at each iteration requires many comparisons between floating-point numbers

to be performed: $O(n^3)$ in total during the overall factorization. Since each comparison costs roughly the same as an arithmetic operation, this overhead is not insignificant.

The more economical pivoting strategy of Bunch and Kaufman [42] searches at most two columns of the working matrix at each stage and requires just $O(n^2)$ comparisons in total. Its rationale and details are somewhat tricky, and we refer the interested reader to the original paper [42] or to Golub and Van Loan [136, Section 4.4] for details. Unfortunately, this algorithm can give rise to arbitrarily large elements in the lower triangular factor L, making it unsuitable for use with a modified Cholesky strategy.

The bounded Bunch–Kaufman strategy is essentially a compromise between the Bunch–Parlett and Bunch–Kaufman strategies. It monitors the sizes of elements in L, accepting the (inexpensive) Bunch–Kaufman choice of pivot block when it yields only modest element growth, but searching further for an acceptable pivot when this growth is excessive. Its total cost is usually similar to that of Bunch–Kaufman, but in the worst case it can approach the cost of Bunch–Parlett.

So far, we have ignored the effect of the choice of pivot block E on the sparsity of the final L factor. This consideration is important when the matrix to be factored is large and sparse, since it greatly affects both the CPU time and the amount of storage required by the algorithm. Algorithms that modify the strategies above to take account of sparsity have been proposed by Duff et al. [97], Duff and Reid [95], and Fourer and Mehrotra [113].

SHERMAN–MORRISON–WOODBURY FORMULA

If the square nonsingular matrix A undergoes a rank-one update to become

$$\bar{A} = A + ab^T,$$

where $a, b \in \mathbb{R}^n$, then if $\bar{A}$ is nonsingular, we have

$$\bar{A}^{-1} = A^{-1} - \frac{A^{-1}ab^TA^{-1}}{1 + b^TA^{-1}a}. \tag{A.27}$$

It is easy to verify this formula: Simply multiply the definitions of $\bar{A}$ and $\bar{A}^{-1}$ together and check that they produce the identity.

This formula can be extended to higher-rank updates. Let U and V be matrices in $\mathbb{R}^{n\times p}$ for some p between 1 and n. If we define

$$\hat{A} = A + UV^T,$$

then $\hat{A}$ is nonsingular if and only if $(I + V^TA^{-1}U)$ is nonsingular, and in this case we have

$$\hat{A}^{-1} = A^{-1} - A^{-1}U(I + V^TA^{-1}U)^{-1}V^TA^{-1}. \tag{A.28}$$

We can use this formula to solve linear systems of the form $\bar{A}x = d$. Since

$$x = \hat{A}^{-1}d = A^{-1}d - A^{-1}U(I + V^T A^{-1}U)^{-1}V^T A^{-1}d,$$

we see that x can be found by solving $p+1$ linear systems with the matrix A (to obtain $A^{-1}d$ and $A^{-1}U$), inverting the $p \times p$ matrix $I + V^T A^{-1}U$, and performing some elementary matrix algebra. Inversion of the $p \times p$ matrix $I + V^T A^{-1}U$ is inexpensive when $p \ll n$.

INTERLACING EIGENVALUE THEOREM

The following result is proved for example in Golub and Van Loan [136, Theorem 8.1.8].

Theorem A.1 (Interlacing Eigenvalue Theorem).
Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ satisfying

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n,$$

and let $z \in \mathbb{R}^n$ be a vector with $\|z\| = 1$, and $\alpha \in \mathbb{R}$ be a scalar. Then if we denote the eigenvalues of $A + \alpha zz^T$ by $\xi_1, \xi_2, \ldots, \xi_n$ (in decreasing order), we have for $\alpha > 0$ that

$$\xi_1 \geq \lambda_1 \geq \xi_2 \geq \lambda_2 \geq \xi_3 \geq \cdots \geq \xi_n \geq \lambda_n,$$

with

$$\sum_{i=1}^{n} \xi_i - \lambda_i = \alpha. \tag{A.29}$$

If $\alpha < 0$, we have that

$$\lambda_1 \geq \xi_1 \geq \lambda_2 \geq \xi_2 \geq \lambda_3 \geq \cdots \geq \lambda_n \geq \xi_n,$$

where the relationship (A.29) is again satisfied.

Informally stated, the eigenvalues of the modified matrix "interlace" the eigenvalues of the original matrix, with nonnegative adjustments if the coefficient α is positive, and nonpositive adjustments if α is negative. The total magnitude of the adjustments equals α, whose magnitude is identical to the Euclidean norm $\|\alpha zz^T\|_2$ of the modification.

ERROR ANALYSIS AND FLOATING-POINT ARITHMETIC

In most of this book our algorithms and analysis deal with real numbers. Modern digital computers, however, cannot store or compute with general real numbers. Instead,

they work with a subset known as *floating-point numbers*. Any quantities that are stored on the computer, whether they are read directly from a file or program or arise as the intermediate result of a computation, must be approximated by a floating-point number. In general, then, the numbers that are produced by practical computation differ from those that would be produced if the arithmetic were exact. Of course, we try to perform our computations in such a way that these differences are as tiny as possible.

Discussion of errors requires us to distinguish between *absolute error* and *relative error*. If x is some exact quantity (scalar, vector, matrix) and $\tilde{x}$ is its approximate value, the absolute error is the norm of the difference, namely, $\|x - \tilde{x}\|$. (In general, any of the norms (A.2a), (A.2b), and (A.2c) can be used in this definition.) The relative error is the ratio of the absolute error to the size of the exact quantity, that is,

$$\frac{\|x - \tilde{x}\|}{\|x\|}.$$

When this ratio is significantly less than one, we can replace the denominator by the size of the approximate quantity—that is, $\|\tilde{x}\|$—without affecting its value very much.

Most computations associated with optimization algorithms are performed in double-precision arithmetic. Double-precision numbers are stored in words of length 64 bits. Most of these bits (say t) are devoted to storing the *fractional part*, while the remainder encode the *exponent* e and other information, such as the sign of the number, or an indication of whether it is zero or "undefined." Typically, the fractional part has the form

$$.d_1 d_2 \ldots d_t,$$

where each $d_i, i = 1, 2, \ldots, t$, is either zero or one. (In some systems d_1 is implicitly assumed to be 1 and is not stored.) The value of the floating-point number is then

$$\sum_{i=1}^{t} d_i 2^{-i} \times 2^e.$$

The value 2^{-t-1} is known as *unit roundoff* and is denoted by $\mathbf{u}$. Any real number whose absolute value lies in the range $[2^L, 2^U]$ (where L and U are lower and upper bounds on the value of the exponent e) can be approximated to within a relative accuracy of $\mathbf{u}$ by a floating-point number, that is,

$$\text{fl}(x) = x(1 + \epsilon), \qquad \text{where } |\epsilon| \leq \mathbf{u}, \tag{A.30}$$

where $\text{fl}(\cdot)$ denotes floating-point approximation. The value of $\mathbf{u}$ for double-precision IEEE arithmetic is about 1.1×10^{-16}. In other words, if the real number x and its floating-point approximation are both written as base-10 numbers (the usual fashion), they agree to at least 15 digits.

For further information on floating-point computations, see Overton [233], Golub and Van Loan [136, Section 2.4], and Higham [169].

When an arithmetic operation is performed with one or two floating-point numbers, the result must also be stored as a floating-point number. This process introduces a small *roundoff error*, whose size can be quantified in terms of the size of the arguments. If x and y are two floating-point numbers, we have that

$$|\mathrm{fl}(x * y) - x * y| \leq \mathbf{u}|x * y|, \tag{A.31}$$

where $*$ denotes any of the operations $+, -, \times, \div$.

Although the error in a single floating-point operation appears benign, more significant errors may occur when the arguments x and y are floating-point approximations of two *real* numbers, or when a sequence of computations are performed in succession. Suppose, for instance, that x and y are large real numbers whose values are very similar. When we store them in a computer, we approximate them with floating-point numbers $\mathrm{fl}(x)$ and $\mathrm{fl}(y)$ that satisfy

$$\mathrm{fl}(x) = x + \epsilon_x, \quad \mathrm{fl}(y) = y + \epsilon_y, \quad \text{where } |\epsilon_x| \leq \mathbf{u}|x|, |\epsilon_y| \leq \mathbf{u}|y|.$$

If we take the difference of the two stored numbers, we obtain a final result $\mathrm{fl}(\mathrm{fl}(x) - \mathrm{fl}(y))$ that satisfies

$$\mathrm{fl}(\mathrm{fl}(x) - \mathrm{fl}(y)) = (\mathrm{fl}(x) - \mathrm{fl}(y))(1 + \epsilon_{xy}), \quad \text{where } |\epsilon_{xy}| \leq \mathbf{u}.$$

By combining these expressions, we find that the difference between this result and the true value $x - y$ may be as large as

$$\epsilon_x + \epsilon_y + \epsilon_{xy},$$

which is bounded by $\mathbf{u}(|x| + |y| + |x - y|)$. Hence, since x and y are large and close together, the relative error is approximately $2\mathbf{u}|x|/|x - y|$, which may be quite large, since $|x| \gg |x - y|$.

This phenomenon is known as *cancellation*. It can also be explained (less formally) by noting that if both x and y are accurate to k digits, and if they agree in the first $\bar{k}$ digits, then their difference will contain only about $k - \bar{k}$ significant digits—the first $\bar{k}$ digits cancel each other out. This observation is the reason for the well-known adage of numerical computing—that one should avoid taking the difference of two similar numbers if at all possible.

CONDITIONING AND STABILITY

Conditioning and *stability* are two terms that are used frequently in connection with numerical computations. Unfortunately, their meaning sometimes varies from author to author, but the general definitions below are widely accepted, and we adhere to them in this book.

Conditioning is a property of the numerical problem at hand (whether it is a linear algebra problem, an optimization problem, a differential equations problem, or whatever). A problem is said to be *well conditioned* if its solution is not affected greatly by small perturbations to the data that define the problem. Otherwise, it is said to be *ill conditioned*.

A simple example is given by the following 2×2 system of linear equations:

$$\begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}.$$

By computing the inverse of the coefficient matrix, we find that the solution is simply

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 & 2 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

If we replace the first right-hand-side element by 3.00001, the solution becomes $(x_1, x_2)^T = (0.99999, 1.00001)^T$, which is only slightly different from its exact value $(1, 1)^T$. We would note similar insensitivity if we were to perturb the other elements of the right-hand-side or elements of the coefficient matrix. We conclude that this problem is well conditioned. On the other hand, the problem

$$\begin{bmatrix} 1.00001 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2.00001 \\ 2 \end{bmatrix}$$

is ill conditioned. Its exact solution is $x = (1, 1)^T$, but if we change the first element of the right-hand-side from 2.00001 to 2, the solution would change drastically to $x = (0, 2)^T$.

For general square linear systems $Ax = b$ where $A \in \mathbb{R}^{n \times n}$, the condition number of the matrix (defined in (A.11)) can be used to quantify the conditioning. Specifically, if we perturb A to $\tilde{A}$ and b to $\tilde{b}$ and take $\tilde{x}$ to be the solution of the perturbed system $\tilde{A}\tilde{x} = \tilde{b}$, it can be shown that

$$\frac{\|x - \tilde{x}\|}{\|x\|} \approx \kappa(A) \left[\frac{\|A - \tilde{A}\|}{\|A\|} + \frac{\|b - \tilde{b}\|}{\|b\|} \right]$$

(see, for instance, Golub and Van Loan [136, Section 2.7]). Hence, a large condition number $\kappa(A)$ indicates that the problem $Ax = b$ is ill conditioned, while a modest value indicates well conditioning.

Note that the concept of conditioning has nothing to do with the particular algorithm that is used to solve the problem, only with the numerical problem itself.

Stability, on the other hand, is a property of the algorithm. An algorithm is stable if it is guaranteed to produce accurate answers to all well-conditioned problems in its class, even when floating-point arithmetic is used.

As an example, consider again the linear equations $Ax = b$. We can show that Algorithm A.1, in combination with triangular substitution, yields a computed solution $\tilde{x}$ whose relative error is approximately

$$\frac{\|x - \tilde{x}\|}{\|x\|} \approx \kappa(A) \frac{\text{growth}(A)}{\|A\|} \mathbf{u}, \tag{A.32}$$

where growth(A) is the size of the largest element that arises in A during execution of Algorithm A.1. In the worst case, we can show that growth$(A)/\|A\|$ may be around 2^{n-1}, which indicates that Algorithm A.1 is an unstable algorithm, since even for modest n (say, $n = 200$), the right-hand-side of (A.32) may be large even when $\kappa(A)$ is modest. In practice, however, large growth factors are rarely observed, so we conclude that Algorithm A.1 is stable for all practical purposes.

Gaussian elimination without pivoting, on the other hand, is definitely unstable. If we omit the possible exchange of rows in Algorithm A.1, the algorithm will fail to produce a factorization even of some well-conditioned matrices, such as

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

For systems $Ax = b$ in which A is symmetric positive definite, the Cholesky factorization in combination with triangular substitution constitutes a stable algorithm for producing a solution x.

A.2 ELEMENTS OF ANALYSIS, GEOMETRY, TOPOLOGY

SEQUENCES

Suppose that $\{x_k\}$ is a sequence of points belonging to $\mathbb{R}^n$. We say that a sequence $\{x_k\}$ *converges* to some point x, written $\lim_{k\to\infty} x_k = x$, if for any $\epsilon > 0$, there is an index K such that

$$\|x_k - x\| \leq \epsilon, \quad \text{for all } k \geq K.$$

For example, the sequence $\{x_k\}$ defined by $x_k = (1 - 2^{-k}, 1/k^2)^T$ converges to $(1, 0)^T$.

Given a index set $\mathcal{S} \subset \{1, 2, 3, \ldots\}$, we can define a *subsequence* of $\{t_k\}$ corresponding to $\mathcal{S}$, and denote it by $\{t_k\}_{k\in\mathcal{S}}$.

We say that $\hat{x} \in \mathbb{R}^n$ is an *accumulation point* or *limit point* for $\{x_k\}$ if there is an infinite set of indices $k_1, k_2, k_3, \ldots$ such that the subsequence $\{x_{k_i}\}_{i=1,2,3,\ldots}$ converges to $\hat{x}$; that is,

$$\lim_{i\to\infty} x_{k_i} = \hat{x}.$$

Alternatively, we say that for any $\epsilon > 0$ and all positive integers K, we have

$$\|x_k - x\| \leq \epsilon, \quad \text{for some } k \geq K.$$

An example is given by the sequence

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1/8 \\ 1/8 \end{bmatrix}, \ldots, \tag{A.33}$$

which has exactly two limit points: $\hat{x} = (0, 0)^T$ and $\hat{x} = (1, 1)^T$. A sequence can even have an infinite number of limit points. An example is the sequence $x_k = \sin k$, for which every point in the interval $[-1, 1]$ is a limit point. A sequence converges if and only if it has exactly one limit point.

A sequence is said to be a Cauchy sequence if for any $\epsilon > 0$, there exists an integer K such that $\|x_k - x_l\| \leq \epsilon$ for all indices $k \geq K$ and $l \geq K$. A sequence converges if and only if it is a Cauchy sequence.

We now consider *scalar sequences* $\{t_k\}$, that is, $t_k \in \mathbb{R}$ for all k. This sequence is said to be *bounded above* if there exists a scalar u such that $t_k \leq u$ for all k, and bounded below if there is a scalar v with $t_k \geq v$ for all k. The sequence $\{t_k\}$ is said to be *nondecreasing* if $t_{k+1} \geq t_k$ for all k, and *nonincreasing* if $t_{k+1} \leq t_k$ for all k. If $\{t_k\}$ is *nondecreasing and bounded above*, then it converges, that is, $\lim_{k\to\infty} t_k = t$ for some scalar t. Similarly, if $\{t_k\}$ is nonincreasing and bounded below, it converges.

We define the *supremum* of the scalar sequence $\{t_k\}$ as the smallest real number u such that $t_k \leq u$ for all $k = 1, 2, 3, \ldots$, and denote it by $\sup\{t_k\}$. The *infimum*, denoted by $\inf\{t_k\}$, is the largest real number v such that $v \leq t_k$ for all $k = 1, 2, 3, \ldots$. We can now define the sequence of suprema as $\{u_i\}$, where

$$u_i \stackrel{\text{def}}{=} \sup\{t_k \mid k \geq i\}.$$

Clearly, $\{u_i\}$ is a nonincreasing sequence. If bounded below, it converges to a finite number $\bar{u}$, which we call the "lim sup" of $\{t_k\}$, denoted by $\limsup t_k$. Similarly, we can denote the sequence of infima by $\{v_i\}$, where

$$v_i \stackrel{\text{def}}{=} \inf\{t_k \mid k \geq i\},$$

which is nondecreasing. If $\{v_i\}$ is bounded above, it converges to a point $\bar{v}$ which we call the "lim inf" of $\{t_k\}$, denoted by $\liminf t_k$. As an example, the sequence $1, \frac{1}{2}, 1, \frac{1}{4}, 1, \frac{1}{8}, \ldots$ has a lim inf of 0 and a lim sup of 1.

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

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

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

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

For example, the sequence $1 + k^{-k}$ converges superlinearly to 1. (Prove this statement!) *Q-quadratic* convergence, an even more rapid convergence rate, is obtained if

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} \leq M, \quad \text{for all } k \text{ sufficiently large,}$$

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

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

Quasi-Newton methods for unconstrained optimization typically converge Q-superlinearly, whereas Newton's method converges Q-quadratically under appropriate assumptions. In contrast, steepest descent algorithms converge only at a Q-linear rate, and when the problem is ill-conditioned the convergence constant r in (A.34) is close to 1.

In the book, we omit the letter Q and simply talk about superlinear convergence, quadratic convergence, and so on.

A slightly weaker form of convergence, characterized by the prefix "R" (for "root"), is concerned with the overall rate of decrease in the error, rather than the decrease over each individual step of the algorithm. We say that convergence is *R-linear* if there is a sequence of nonnegative scalars $\{v_k\}$ such that

$$\|x_k - x^*\| \leq v_k \text{ for all } k, \text{ and } \{v_k\} \text{ converges Q-linearly to zero.}$$

The sequence $\{\|x_k - x^*\|\}$ is said to be *dominated* by $\{v_k\}$. For instance, the sequence

$$x_k = \begin{cases} 1 + (0.5)^k, & k \text{ even}, \\ 1, & k \text{ odd}, \end{cases} \tag{A.35}$$

(the first few iterates are 2, 1, 1.25, 1, 1.03125, 1, ...) converges R-linearly to 1, because we have $(1 + (0.5)^k) - 1| = (0.)^k$, and the sequence $\{(0.5)^k\}$ converges Q-linearly to zero. Likewise, we say that $\{x_k\}$ converges R-superlinearly to x^* if $\{\|x_k - x^*\|\}$ is dominated by a sequence of scalars converging Q-superlinearly to zero, and $\{x_k\}$ converges R-quadratically to x^* if $\{\|x_k - x^*\|\}$ is dominated by a sequence converging Q-quadratically to zero.

Note that in the R-linear sequence (A.35), the error actually increases at every second iteration! Such behavior occurs even in sequences whose R-rate of convergence is arbitrarily high, but it cannot occur for Q-linear sequences, which insist on a decrease at every step k, for k sufficiently large.

For an extensive discussion of convergence rates see Ortega and Rheinboldt [230].

TOPOLOGY OF THE EUCLIDEAN SPACE $\mathbb{R}^n$

The set $\mathcal{F}$ is *bounded* if there is some real number $M > 0$ such that

$$\|x\| \leq M, \quad \text{for all } x \in \mathcal{F}.$$

A subset $\mathcal{F} \subset \mathbb{R}^n$ is *open* if for every $x \in \mathcal{F}$, we can find a positive number $\epsilon > 0$ such that the ball of radius ϵ around x is contained in $\mathcal{F}$; that is,

$$\{y \in \mathbb{R}^n \mid \|y - x\| \leq \epsilon\} \subset \mathcal{F}.$$

The set $\mathcal{F}$ is *closed* if for all possible sequences of points $\{x_k\}$ in $\mathcal{F}$, all limit points of $\{x_k\}$ are elements of $\mathcal{F}$. For instance, the set $\mathcal{F} = (0, 1) \cup (2, 10)$ is an open subset of $\mathbb{R}$, while

$\mathcal{F} = [0, 1] \cup [2, 5]$ is a closed subset of $\mathbb{R}$. The set $\mathcal{F} = (0, 1]$ is a subset of $\mathbb{R}$ that is neither open nor closed.

The *interior* of a set $\mathcal{F}$, denoted by int $\mathcal{F}$, is the largest open set contained in $\mathcal{F}$. The *closure* of $\mathcal{F}$, denoted by cl $\mathcal{F}$, is the smallest closed set containing $\mathcal{F}$. In other words, we have

$$x \in \text{cl}\mathcal{F} \quad \text{if } \lim_{k\to\infty} x_k = x \text{ for some sequence } \{x_k\} \text{ of points in } \mathcal{F}.$$

If $\mathcal{F} = (-1, 1] \cup [2, 4)$, then

$$\text{cl}\mathcal{F} = [-1, 1] \cup [2, 4], \qquad \text{int } \mathcal{F} = (-1, 1) \cup (2, 4).$$

Note that if $\mathcal{F}$ is open, then int $\mathcal{F} = \mathcal{F}$, while if $\mathcal{F}$ is closed, then cl $\mathcal{F} = \mathcal{F}$.

We note the following facts about open and closed sets. The union of finitely many closed sets is closed, while any intersection of closed sets is closed. The intersection of finitely many open sets is open, while any union of open sets is open.

The set $\mathcal{F}$ is *compact* if every sequence $\{x^k\}$ of points in $\mathcal{F}$ has at least one limit point, and all such limit points are in $\mathcal{F}$. (This definition is equivalent to the more formal one involving covers of $\mathcal{F}$.) The following is a central result in topology:

$\mathcal{F} \in \mathbb{R}^n$ is closed and bounded $\Rightarrow \mathcal{F}$ is compact.

Given a point $x \in \mathbb{R}^n$, we call $\mathcal{N} \in \mathbb{R}^n$ a *neighborhood of* x if it is an open set containing x. An especially useful neighborhood is the *open ball of radius* ϵ *around* x, which is denoted by $\mathbb{B}(x, \epsilon)$; that is,

$$\mathbb{B}(x, \epsilon) = \{y \mid \|y - x\| < \epsilon\}.$$

Given a set $\mathcal{F} \subset \mathbb{R}^n$, we say that $\mathcal{N}$ is a *neighborhood of* $\mathcal{F}$ if there is $\epsilon > 0$ such that

$$\cup_{x\in\mathcal{F}} \mathbb{B}(x, \epsilon) \subset \mathcal{N}.$$

CONVEX SETS IN $\mathbb{R}^n$

A *convex combination* of a finite set of vectors $\{x_1, x_2, \ldots, x_m\}$ in $\mathbb{R}^m$ is any vector x of the form

$$x = \sum_{i=1}^{m} \alpha_i x_i, \quad \text{where } \sum_{i=1}^{m} \alpha_i = 1, \quad \text{and } \alpha_i \geq 0 \text{ for all } i = 1, 2, \ldots, m.$$

The *convex hull* of $\{x_1, x_2, \ldots, x_m\}$ is the set of all convex combinations of these vectors.

A *cone* is a set $\mathcal{F}$ with the property that for all $x \in \mathcal{F}$ we have

$$x \in \mathcal{F} \;\Rightarrow\; \alpha x \in \mathcal{F}, \quad \text{for all } \alpha > 0. \tag{A.36}$$

For instance, the set $\mathcal{F} \subset \mathbb{R}^2$ defined by

$$\{(x_1, x_2)^T \mid x_1 > 0,\ x_2 \geq 0\}$$

is a cone in $\mathbb{R}^2$. Note that cones are not necessarily convex. For example, the set $\{(x_1, x_2)^T \mid x_1 \geq 0 \text{ or } x_2 \geq 0\}$, which encompasses three quarters of the two-dimensional plane, is a cone.

The *cone generated by* $\{x_1, x_2, \ldots, x_m\}$ is the set of all vectors x of the form

$$x = \sum_{i=1}^{m} \alpha_i x_i, \quad \text{where } \alpha_i \geq 0 \text{ for all } i = 1, 2, \ldots, m.$$

Note that all cones of this form are convex.

Finally, we define the affine hull and relative interior of a set. An *affine set* in $\mathbb{R}^n$ is a the set of all vectors $\{x\} \oplus \mathcal{S}$, where $x \in \mathbb{R}^n$ and $\mathcal{S}$ is a subspace of $\mathbb{R}^n$. Given $\mathcal{F} \subset \mathbb{R}^n$, the *affine hull* of $\mathcal{F}$ (denoted by $\text{aff}\,\mathcal{F}$) is the smallest affine set containing $\mathcal{F}$. For instance, when $\mathcal{F}$ is the "ice-cream cone" defined in three dimensions as

$$\Omega = \left\{ x \in \mathbb{R}^3 \mid x_3 \geq 2\sqrt{x_1^2 + x_2^2} \right\} \tag{A.37}$$

(see Figure A.1), we have $\text{aff}\,\mathcal{F} = \mathbb{R}^3$. If $\mathcal{F}$ is the set of two isolated points $\mathcal{F} = \{(1, 0, 0)^T, (0, 2, 0)^T\}$, we have

$$\text{aff}\,\mathcal{F} = \{(1, 0, 0)^T + \alpha(-1, 2, 0)^T \mid \text{for all } \alpha \in \mathbb{R}\}.$$

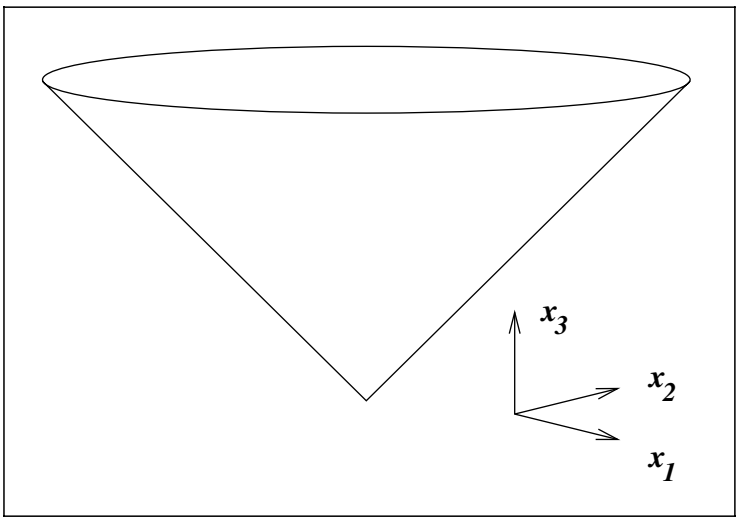

Figure A.1 "Ice-cream cone" set.

The *relative interior* $\operatorname{ri}\mathcal{F}$ of the set $\mathcal{F}$ is its *interior relative to* $\operatorname{aff}\mathcal{F}$. If $x \in \mathcal{F}$, then $x \in \operatorname{ri}\mathcal{F}$ if there is an $\epsilon > 0$ such that

$$(x + \epsilon\mathcal{B}) \cap \operatorname{aff}\mathcal{F} \subset \mathcal{F}.$$

Referring again to the ice-cream cone (A.37), we have that

$$\operatorname{ri}\mathcal{F} = \left\{ x \in \mathbb{R}^3 \;\middle|\; x_3 > 2\sqrt{x_1^2 + x_2^2} \right\}.$$

For the set of two isolated points $\mathcal{F} = \{(1, 0, 0)^T, (0, 2, 0)^T\}$, we have $\operatorname{ri}\mathcal{F} = \emptyset$. For the set $\mathcal{F}$ defined by

$$\mathcal{F} \stackrel{\text{def}}{=} \{x \in \mathbb{R}^3 \,|\, x_1 \in [0, 1],\ x_2 \in [0, 1],\ x_3 = 0\},$$

we have that

$$\operatorname{aff}\mathcal{F} = \mathbb{R} \times \mathbb{R} \times \{0\}, \qquad \operatorname{ri}\mathcal{F} = \{x \in \mathbb{R}^3 \,|\, x_1 \in (0, 1),\ x_2 \in (0, 1),\ x_3 = 0\}.$$

CONTINUITY AND LIMITS

Let f be a function that maps some domain $\mathcal{D} \subset \mathbb{R}^n$ to the space $\mathbb{R}^m$. For some point $x_0 \in \operatorname{cl}\mathcal{D}$, we write

$$\lim_{x \to x_0} f(x) = f_0 \tag{A.38}$$

(spoken "the limit of $f(x)$ as x approaches x_0 is f_0") if for all $\epsilon > 0$, there is a value $\delta > 0$ such that

$$\|x - x_0\| < \delta \ \text{ and } \ x \in \mathcal{D} \ \Rightarrow \ \|f(x) - f_0\| < \epsilon.$$

We say that f is *continuous* at x_0 if $x_0 \in \mathcal{D}$ and the expression (A.38) holds with $f_0 = f(x_0)$. We say that f is continuous on its domain $\mathcal{D}$ if f is continuous for all $x_0 \in \mathcal{D}$.

An example is provided by the function

$$f(x) = \begin{cases} -x & \text{if } x \in [-1, 1], x \neq 0, \\ 5 & \text{for all other } x \in [-10, 10]. \end{cases} \tag{A.39}$$

This function is defined on the domain $[-10, 10]$ and is continuous at all points of the domain except the points $x = 0$, $x = 1$, and $x = -1$. At $x = 0$, the expression (A.38) holds with $f_0 = 0$, but the function is not continuous at this point because $f_0 \neq f(0) = 5$. At

$x = -1$, the limit (A.38) is not defined, because the function values in the neighborhood of this point are close to both 5 and -1, depending on whether x is slightly smaller or slightly larger than -1. Hence, the function is certainly not continuous at this point. The same comments apply to the point $x = 1$.

In the special case of $n = 1$ (that is, the argument of f is a real scalar), we can also define the *one-sided limit*. Given $x_0 \in \text{cl}\mathcal{D}$, We write

$$\lim_{x \downarrow x_0} f(x) = f_0 \tag{A.40}$$

(spoken "the limit of $f(x)$ as x approaches x_0 from above is f_0") if for all $\epsilon > 0$, there is a value $\delta > 0$ such that

$$x_0 < x < x_0 + \delta \;\text{ and }\; x \in \mathcal{D} \;\;\Rightarrow\;\; \|f(x) - f_0\| < \epsilon.$$

Similarly, we write

$$\lim_{x \uparrow x_0} f(x) = f_0 \tag{A.41}$$

(spoken "the limit of $f(x)$ as x approaches x_0 from below is f_0") if for all $\epsilon > 0$, there is a value $\delta > 0$ such that

$$x_0 - \delta < x < x_0 \;\text{ and }\; x \in \mathcal{D} \;\;\Rightarrow\;\; \|f(x) - f_0\| < \epsilon.$$

For the function defined in (A.39), we have that

$$\lim_{x \downarrow 1} f(x) = 5, \qquad \lim_{x \uparrow 1} f(x) = 1.$$

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 $x \in \mathcal{N}$). We prove this claim via a sequence of elementary inequalities, for arbitrary $x_0, x_1 \in \mathcal{N}$:

$$\begin{aligned}
&|g(x_0)h(x_0) - g(x_1)h(x_1)| \\
&\leq |g(x_0)h(x_0) - g(x_1)h(x_0)| + |g(x_1)h(x_0) - g(x_1)h(x_1)| \\
&= |h(x_0)|\,|g(x_0) - g(x_1)| + |g(x_1)|\,|h(x_0) - h(x_1)| \\
&\leq 2ML\|x_0 - x_1\|, \qquad \text{(A.43)}
\end{aligned}$$

where L is an upper bound on the Lipschitz constant for both g and h.

DERIVATIVES

Let $\phi : \mathbb{R} \to \mathbb{R}$ be a real-valued function of a real variable (sometimes known as a *univariate* function). The first derivative $\phi'(\alpha)$ is defined by

$$\frac{d\phi}{d\alpha} = \phi'(\alpha) \stackrel{\text{def}}{=} \lim_{\epsilon \to 0} \frac{\phi(\alpha + \epsilon) - \phi(\alpha)}{\epsilon}. \qquad \text{(A.44)}$$

The second derivative is obtained by substituting ϕ by ϕ' in this same formula; that is,

$$\frac{d^2\phi}{d\alpha^2} = \phi''(\alpha) \stackrel{\text{def}}{=} \lim_{\epsilon \to 0} \frac{\phi'(\alpha + \epsilon) - \phi'(\alpha)}{\epsilon}. \qquad \text{(A.45)}$$

Suppose now that α in turn depends on another quantity β (we denote this dependence by writing $\alpha = \alpha(\beta)$). We can use the *chain rule* to calculate the derivative of ϕ with respect to β:

$$\frac{d\phi(\alpha(\beta))}{d\beta} = \frac{d\phi}{d\alpha}\frac{d\alpha}{d\beta} = \phi'(\alpha)\alpha'(\beta). \qquad \text{(A.46)}$$

Consider now the function $f : \mathbb{R}^n \to \mathbb{R}$, which is a real-valued function of n independent variables. We typically gather the variables into a vector $x = (x_1, x_2, \ldots, x_n)^T$. We say that f is differentiable at x if there exists a vector $g \in \mathbb{R}^n$ such that

$$\lim_{y \to 0} \frac{f(x + y) - f(x) - g^T y}{\|y\|} = 0, \qquad \text{(A.47)}$$

where $\|\cdot\|$ is any vector norm of y. (This type of differentiability is known as *Frechet differentiability*.) If g satisfying (A.47) exists, we call it the *gradient* of f at x, and denote it

by $\nabla f(x)$, written componentwise as

$$\nabla f(x) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix}. \tag{A.48}$$

Here, $\partial f/\partial x_i$ represents the partial derivative of f with respect to x_i. By setting $y = \epsilon e_i$ in (A.47), where e_i is the vector in $\mathbb{R}^n$ consisting all all zeros, except for a 1 in position i, we obtain

$$\begin{aligned} &\frac{\partial f}{\partial x_i} \\ &\stackrel{\text{def}}{=} \lim_{\epsilon \to 0} \frac{f(x_1, \ldots, x_{i-1}, x_i + \epsilon, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)}{\epsilon} \\ &= \frac{f(x + \epsilon e_i) - f(x)}{\epsilon}. \end{aligned}$$

A gradient with respect to only a subset of the unknowns can be expressed by means of a subscript on the symbol ∇. Thus for the function of two vector variables $f(z, t)$, we use $\nabla_z f(z, t)$ to denote the gradient with respect to z (holding t constant).

The matrix of second partial derivatives of f is known as the *Hessian*, and is defined as

$$\nabla^2 f(x) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

We say that f is *differentiable* on a domain $\mathcal{D}$ if $\nabla f(x)$ exists for all $x \in \mathcal{D}$, and *continuously differentiable* if $\nabla f(x)$ is a continuous functions of x. Similarly, f is *twice differentiable* on $\mathcal{D}$ if $\nabla^2 f(x)$ exists for all $x \in \mathcal{D}$ and *twice continuously differentiable* if $\nabla^2 f(x)$ is continuous on $\mathcal{D}$. Note that when f is twice continuously differentiable, the Hessian is a symmetric matrix, since

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}, \quad \text{for all } i, j = 1, 2, \ldots, n.$$

When f is a vector valued function that is $f : \mathbb{R}^n \to \mathbb{R}^m$ (See Chapters 10 and 11), we define $\nabla f(x)$ to be the $n \times m$ matrix whose ith column is $\nabla f_i(x)$, that is, the gradient of

f_i with respect to x. Often, for notational convenience, we prefer to work with the transpose of his matrix, which has dimensions $m \times n$. This matrix is called the *Jacobian* and is often denoted by $J(x)$. Specifically, the (i, j) element of $J(x)$ is $\partial f_i(x)/\partial x_j$.

When the vector x in turn depends on another vector t (that is, $x = x(t)$), we can extend the chain rule (A.46) for the univariate function. Defining

$$h(t) = f(x(t)), \tag{A.49}$$

we have

$$\nabla h(t) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \nabla x_i(t) = \nabla x(t) \nabla f(x(t)). \tag{A.50}$$

❑ EXAMPLE A.1

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be defined by $f(x_1, x_2) = x_1^2 + x_1 x_2$, where $x_1 = \sin t_1 + t_2^2$ and $x_2 = (t_1 + t_2)^2$. Defining $h(t)$ as in (A.49), the chain rule (A.50) yields

$$\begin{aligned}
&\nabla h(t) \\
&\quad = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \nabla x_i(t) \\
&\quad = (2x_1 + x_2) \begin{bmatrix} \cos t_1 \\ 2t_2 \end{bmatrix} + x_1 \begin{bmatrix} 2(t_1 + t_2) \\ 2(t_1 + t_2) \end{bmatrix} \\
&\quad = \left(2\left(\sin t_1 + t_2^2\right) + (t_1 + t_2)^2\right) \begin{bmatrix} \cos t_1 \\ 2t_2 \end{bmatrix} + \left(\sin t_1 + t_2^2\right) \begin{bmatrix} 2(t_1 + t_2) \\ 2(t_1 + t_2) \end{bmatrix}.
\end{aligned}$$

If, on the other hand, we substitute directly for x into the definition of f, we obtain

$$h(t) = f(x(t)) = \left(\sin t_1 + t_2^2\right)^2 + \left(\sin t_1 + t_2^2\right)(t_1 + t_2)^2.$$

The reader should verify that the gradient of this expression is identical to the one obtained above by applying the chain rule.

❑

Special cases of the chain rule can be derived when $x(t)$ in (A.50) is a linear function of t, say $x(t) = Ct$. We then have $\nabla x(t) = C^T$, so that

$$\nabla h(t) = C^T \nabla f(Ct).$$

In the case in which f is a scalar function, we can differentiate twice using the chain rule to obtain

$$\nabla^2 h(t) = C^T \nabla^2 f(Ct) C.$$

(The proof of this statement is left as an exercise.)

DIRECTIONAL DERIVATIVES

The *directional derivative* of a function $f : \mathbb{R}^n \to \mathbb{R}$ in the direction p is given by

$$D(f(x); p) \stackrel{\text{def}}{=} \lim_{\epsilon \to 0} \frac{f(x + \epsilon p) - f(x)}{\epsilon}. \tag{A.51}$$

The directional derivative may be well defined even when f is not continuously differentiable; in fact, it is most useful in such situations. Consider for instance the ℓ_1 norm function $f(x) = \|x\|_1$. We have from the definition (A.51) that

$$D(\|x\|_1; p) = \lim_{\epsilon \to 0} \frac{\|x + \epsilon p\|_1 - \|x\|_1}{\epsilon} = \lim_{\epsilon \to 0} \frac{\sum_{i=1}^n |x_i + \epsilon p_i| - \sum_{i=1}^n |x_i|}{\epsilon}.$$

If $x_i > 0$, we have $|x_i + \epsilon p_i| = |x_i| + \epsilon p_i$ for all ϵ sufficiently small. If $x_i < 0$, we have $|x_i + \epsilon p_i| = |x_i| - \epsilon p_i$, while if $x_i = 0$, we have $|x_i + \epsilon p_i| = \epsilon |p_i|$. Therefore, we have

$$D(\|x\|_1; p) = \sum_{i | x_i < 0} -p_i + \sum_{i | x_i > 0} p_i + \sum_{i | x_i = 0} |p_i|,$$

so the directional derivative of this function exists for any x and p. The first derivative $\nabla f(x)$ does *not* exist, however, whenever any of the components of x are zero.

When f is in fact continuously differentiable in a neighborhood of x, we have

$$D(f(x); p) = \nabla f(x)^T p.$$

To verify this formula, we define the function

$$\phi(\alpha) = f(x + \alpha p) = f(y(\alpha)), \tag{A.52}$$

where $y(\alpha) = x + \alpha p$. Note that

$$\lim_{\epsilon \to 0} \frac{f(x + \epsilon p) - f(x)}{\epsilon} = \lim_{\epsilon \to 0} \frac{\phi(\epsilon) - \phi(0)}{\epsilon} = \phi'(0).$$

By applying the chain rule (A.50) to $f(y(\alpha))$, we obtain

$$\begin{aligned}\phi'(\alpha) &= \sum_{i=1}^{n} \frac{\partial f(y(\alpha))}{\partial y_i} \nabla y_i(\alpha) \\ &= \sum_{i=1}^{n} \frac{\partial f(y(\alpha))}{\partial y_i} p_i = \nabla f(y(\alpha))^T p = \nabla f(x+\alpha p)^T p.\end{aligned} \tag{A.53}$$

We obtain (A.51) by setting $\alpha = 0$ and comparing the last two expressions.

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.)

❑ EXAMPLE A.2

Consider $f : \mathbb{R}^2 \to \mathbb{R}$ defined by $f(x) = x_1^3 + 3x_1x_2^2$, and let $x = (0, 0)^T$ and $p = (1, 2)^T$. It is easy to verify that $f(x) = 0$ and $f(x+p) = 13$. Since

$$\nabla f(x+\alpha p) = \begin{bmatrix} 3(x_1+\alpha p_1)^2 + 3(x_2+\alpha p_2)^2 \\ 6(x_1+\alpha p_1)(x_2+\alpha p_2) \end{bmatrix} = \begin{bmatrix} 15\alpha^2 \\ 12\alpha^2 \end{bmatrix},$$

we have that $\nabla f(x+\alpha p)^T p = 39\alpha^2$. Hence the relation (A.55) holds when we set $\alpha = 1/\sqrt{13}$, which lies in the open interval $(0, 1)$, as claimed.

❑

An alternative expression to (A.55) can be stated for twice differentiable functions: We have

$$f(x+p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x+\alpha p)^T p, \tag{A.56}$$

for some $\alpha \in (0, 1)$. In fact, this expression is one form of Taylor's theorem, Theorem 2.1 in Chapter 2, to which we refer throughout the book.

The extension of (A.55) to a vector-valued function $r : \mathbb{R}^n \to \mathbb{R}^m$ for $m > 1$ is not immediate. There is in general no scalar α such that the natural extension of (A.55) is satisfied. However, the following result is often a useful analog. As in (10.3), we denote the Jacobian of $r(x)$, by $J(x)$, where $J(x)$ is the $m \times n$ matrix whose (j, i) entry is $\partial r_j / \partial x_i$, for $j = 1, 2, \ldots, m$ and $i = 1, 2, \ldots, n$, and asssume that $J(x)$ is defined and continuous on the domain of interest. Given x and p, we then have

$$r(x+p) - r(x) = \int_0^1 J(x+\alpha p) p \, d\alpha. \tag{A.57}$$

When p is sufficiently small in norm, we can approximate the right-hand side of this expression adequately by $J(x)p$, that is,

$$r(x+p) - r(x) \approx J(x)p.$$

If J is Lipschitz continuous in the vicinity of x and $x + p$ with Lipschitz constant L, we can use (A.12) to estimate the error in this approximation as follows:

$$\begin{aligned} \|r(x+p) - r(x) - J(x)p\| &= \left\| \int_0^1 [J(x+\alpha p) - J(x)] p \, d\alpha \right\| \\ &\leq \int_0^1 \|J(x+\alpha p) - J(x)\| \, \|p\| \, d\alpha \\ &\leq \int_0^1 L\alpha \|p\|^2 \, d\alpha = \tfrac{1}{2} L \|p\|^2. \end{aligned}$$

IMPLICIT FUNCTION THEOREM

The implicit function theorem lies behind a number of important results in local convergence theory of optimization algorithms and in the characterization of optimality (see Chapter 12). Our statement of this result is based on Lang [187, p. 131] and Bertsekas [19, Proposition A.25].

Theorem A.2 (Implicit Function Theorem).
Let $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ be a function such that

(i) *$h(z^*, 0) = 0$ for some $z^* \in \mathbb{R}^n$,*

(ii) *the function $h(\cdot, \cdot)$ is continuously differentiable in some neighborhood of $(z^*, 0)$, and*

(iii) *$\nabla_z h(z, t)$ is nonsingular at the point $(z, t) = (z^*, 0)$.*

Then there exist open sets $\mathcal{N}_z \subset \mathbb{R}^n$ and $\mathcal{N}_t \subset \mathbb{R}^m$ containing z^* and 0, respectively, and a continuous function $z : \mathcal{N}_t \to \mathcal{N}_z$ such that $z^* = z(0)$ and $h(z(t), t) = 0$ for all $t \in \mathcal{N}_t$. Further, $z(t)$ is uniquely defined. Finally, if h is p times continuously differentiable with respect to both its arguments for some $p > 0$, then $z(t)$ is also p times continuously differentiable with respect to t, and we have

$$\nabla z(t) = -\nabla_t h(z(t), t)[\nabla_z h(z(t), t)]^{-1}$$

for all $t \in \mathcal{N}_t$.

This theorem is frequently applied to parametrized systems of linear equations, in which z is obtained as the solution of

$$M(t)z = g(t),$$

where $M(\cdot) \in \mathbb{R}^{n \times n}$ has $M(0)$ nonsingular, and $g(\cdot) \in \mathbb{R}^n$. To apply the theorem, we define

$$h(z, t) = M(t)z - g(t).$$

If $M(\cdot)$ and $g(\cdot)$ are continuously differentiable in some neighborhood of 0, the theorem implies that $z(t) = M(t)^{-1}g(t)$ is a continuous function of t in some neighborhood of 0.

ORDER NOTATION

In much of our analysis we are concerned with how the members of a sequence behave *eventually*, that is, when we get far enough along in the sequence. For instance, we might ask whether the elements of the sequence are bounded, or whether they are similar in size to the elements of a corresponding sequence, or whether they are decreasing and, if so, how rapidly. *Order notation* is useful shorthand to use when questions like these are being examined. It saves us defining many constants that clutter up the argument and the analysis.

We will use three varieties of order notation: $O(\cdot)$, $o(\cdot)$, and $\Omega(\cdot)$. Given two nonnegative infinite sequences of scalars $\{\eta_k\}$ and $\{\nu_k\}$, we write

$$\eta_k = O(\nu_k)$$

if there is a positive constant C such that

$$|\eta_k| \leq C|\nu_k|$$

for all k sufficiently large. We write

$$\eta_k = o(\nu_k)$$

if the sequence of ratios $\{\eta_k/\nu_k\}$ approaches zero, that is,

$$\lim_{k\to\infty} \frac{\eta_k}{\nu_k} = 0.$$

Finally, we write

$$\eta_k = \Omega(\nu_k)$$

if there are two constants C_0 and C_1 with $0 < C_0 \leq C_1 < \infty$ such that

$$C_0|\nu_k| \leq |\eta_k| \leq C_1|\nu_k|,$$

that is, the corresponding elements of both sequences stay in the same ballpark for all k. This definition is equivalent to saying that $\eta_k = O(\nu_k)$ and $\nu_k = O(\eta_k)$.

The same notation is often used in the context of quantities that depend continuously on each other as well. For instance, if $\eta(\cdot)$ is a function that maps $\mathbb{R}$ to $\mathbb{R}$, we write

$$\eta(\nu) = O(\nu)$$

if there is a constant C such that $|\eta(\nu)| \leq C|\nu|$ for all $\nu \in \mathbb{R}$. (Typically, we are interested only in values of ν that are either very large or very close to zero; this should be clear from the context. Similarly, we use

$$\eta(\nu) = o(\nu) \tag{A.58}$$

to indicate that the ratio $\eta(\nu)/\nu$ approaches zero either as $\nu \to 0$ or $\nu \to \infty$. (Again, the precise meaning should be clear from the context.)

As a slight variant on the definitions above, we write

$$\eta_k = O(1)$$

to indicate that there is a constant C such that $|\eta_k| \leq C$ for all k, while

$$\eta_k = o(1)$$

indicates that $\lim_{k\to\infty} \eta_k = 0$. We sometimes use vector and matrix quantities as arguments, and in these cases the definitions above are intended to apply to the norms of these quantities. For instance, if $f : \mathbb{R}^n \to \mathbb{R}^n$, we write $f(x) = O(\|x\|)$ if there is a constant $C > 0$ such that $\|f(x)\| \leq C\|x\|$ for all x in the domain of f. Typically, as above, we are interested only in some subdomain of f, usually a small neighborhood of 0. As before, the precise meaning should be clear from the context.

ROOT-FINDING FOR SCALAR EQUATIONS

In Chapter 11 we discussed methods for finding solutions of nonlinear systems of equations $F(x) = 0$, where $F : \mathbb{R}^n \to \mathbb{R}^n$. Here we discuss briefly the case of scalar equations ($n = 1$), for which the algorithm is easy to illustrate. Scalar root-finding is needed in the trust-region algorithms of Chapter 4, for instance. Of course, the general theorems of Chapter 11 can be applied to derive rigorous convergence results for this special case.

The basic step of Newton's method (Algorithm Newton of Chapter 11) in the scalar case is simply

$$p_k = -F(x_k)/F'(x_k), \qquad x_{k+1} \leftarrow x_k + p_k \tag{A.59}$$

(cf. (11.6)). Graphically, such a step involves taking the tangent to the graph of F at the point x_k and taking the next iterate to be the intersection of this tangent with the x axis (see Figure A.2). Clearly, if the function F is nearly linear, the tangent will be quite a good approximation to F itself, so the Newton iterate will be quite close to the true root of F.

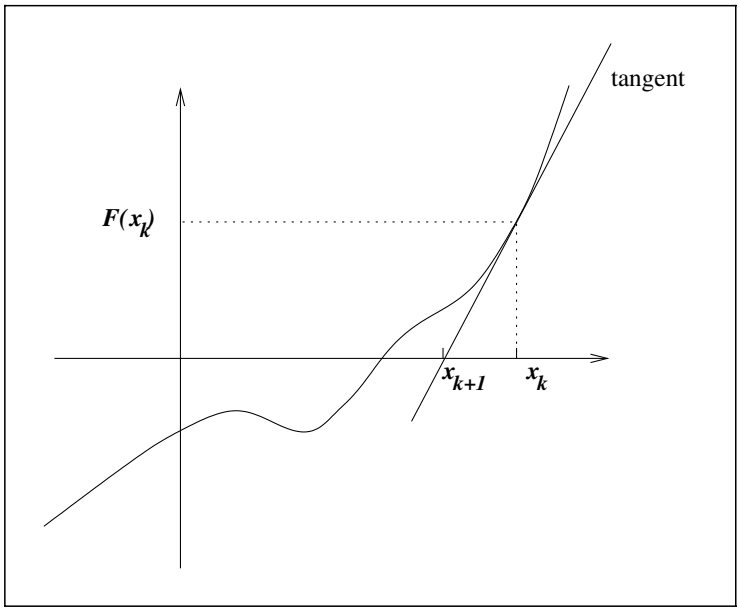

Figure A.2 One step of Newton's method for a scalar equation.

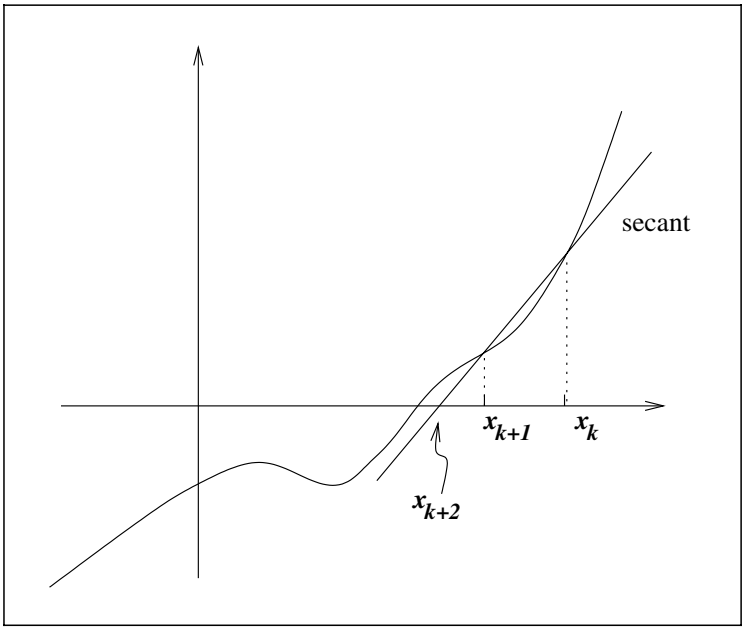

Figure A.3 One step of the secant method for a scalar equation.

The secant method for scalar equations can be viewed as the specialization of Broyden's method to the case of $n = 1$. The issues are simpler in this case, however, since the secant equation (11.27) completely determines the value of the 1×1 approximate Hessian B_k. That is, we do not need to apply extra conditions to ensure that B_k is fully determined. By combining (11.24) with (11.27), we find that the secant method for the case of $n = 1$ is defined by

$$B_k = (F(x_k) - F(x_{k-1}))/(x_k - x_{k-1}), \tag{A.60a}$$

$$p_k = -F(x_k)/B_k, \qquad x_{k+1} = x_k + p_k. \tag{A.60b}$$

By illustrating this algorithm, we see the origin of the term "secant." B_k approximates the slope of the function at x_k by taking the secant through the points $(x_{k-1}, F(x_{k-1})$ and $(x_k, F(x_k))$, and x_{k+1} is obtained by finding the intersection of this secant with the x axis. The method is illustrated in Figure A.3.