CHAPTER 98
Nonlinear Optimization

TOM M. CAVALIER
Pennsylvania State University

1. INTRODUCTION

Nonlinear programming and nonlinear optimization are generally considered synonymous terms, where in this context the term *programming* refers to the process of determining an optimum program or solution. Optimization can be thought of as a very broad extension of the simple calculus problem of finding the extrema of a given function. This process of finding the ''best'' solution to a mathematical model of some real-world system has a wealth of practical applications (see, e.g., Bracken and McCormick 1968.)

The basic elements of a mathematical program are *decision variables,* an *objective function,* and *constraints* or restrictions. To help fix ideas, consider the simple problem of finding the dimensions of a rectangle that has maximum area and whose perimeter is at most 4. This problem can be posed as the following mathematical program:

 Handbook of Industrial Engineering: Technology and Operations Management, Third Edition.
Edited by Gavriel Salvendy

Example 1

$$\text{maximize } f(\mathbf{x}) = x_1x_2 \tag{1}$$

$$\text{subject to } g_1(\mathbf{x}) = 2x_1 + 2x_2 - 4 \leq 0 \tag{2}$$

$$g_2(\mathbf{x}) = -x_1 \leq 0 \tag{3}$$

$$g_3(\mathbf{x}) = -x_2 \leq 0 \tag{4}$$

where $\mathbf{x} = (x_1, x_2)^t \in E_2$, Euclidean 2-space. (The notation $\mathbf{v}^t$ represents the *transpose* of the vector $\mathbf{v}$.) The decision variables in this case are x_1 and x_2, which represent the length and width of the rectangle, respectively. The objective function is $f(\mathbf{x}) = x_1x_2$, which represents the area of the rectangle. $g_1(\mathbf{x}) \leq 0$ is the perimeter constraint, whereas $g_1(\mathbf{x}) \leq 0$ and $g_1(\mathbf{x}) \leq 0$ represent nonnegativity restrictions on the variables. These nonnegativity restrictions, as well as simple bounds on the variables, are handled implicitly by some mathematical programming techniques (e.g., the simplex algorithm for linear programming). Here it is assumed that they are included as explicit constraints.

In general, a mathematical program can be represented in the generic form:

$$\text{(P) optimize } f(\mathbf{x}) \tag{5}$$

$$\text{subject to } g_i(\mathbf{x}) \leq 0 \text{ for } i = 1, \ldots, m \tag{6}$$

$$h_j(\mathbf{x}) = 0 \text{ for } j = 1, \ldots, p \tag{7}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t \in E_n$, Euclidean n-space, and f, g_i, h_j: $E_n \rightarrow E_1$. Generally, throughout much of this chapter, "optimize" will be taken to mean "minimize," since maximization problems can be addressed using the simple transformation, maximize $f(\mathbf{x}) = -$minimize $(-f(\mathbf{x}))$.

If all of the functions f, g_i, h_j are linear functions of $\mathbf{x}$, then (P) is called a *linear program*. Otherwise (P) is a *nonlinear program*. Note that Example 1 is a nonlinear program since the objective function (1) is a nonlinear function. Actually, as will be seen later, this problem can be classified as a *quadratic program* since the objective function is a quadratic function and the constraints are all linear functions.

The region in E_n defined by constraints (6) and (7) is referred to as the *feasible region*; the feasible region for Example 1 is illustrated in Figure 1. The *objective function contours* (or level curves) are also identified in Figure 1, and it is clear that the point, $\mathbf{x}^* = (x_1^*, x_2^*)^t = (1, 1)^t$, where the objective contour is tangent to the boundary of the feasible region, is the global optimal solution. The constraint $g_1(\mathbf{x}) \leq 0$ is said to be *binding* (tight or active) at the optimal solution since $g_1(\mathbf{x}^*) = 0$, whereas $g_2(\mathbf{x}) \leq 0$ and $g_3(\mathbf{x}) \leq 0$ are *nonbinding* (loose or inactive) at the optimal solution.

Unlike linear programming, in which an optimum, if one exists, can be found at an extreme point of the feasible region, solutions of nonlinear programming problems can occur at any feasible point. Whereas Figure 1 shows an example where the optimum lies on the boundary of the feasible region, Figure 2 illustrates a case where the optimum is an interior point of the feasible region. In the latter

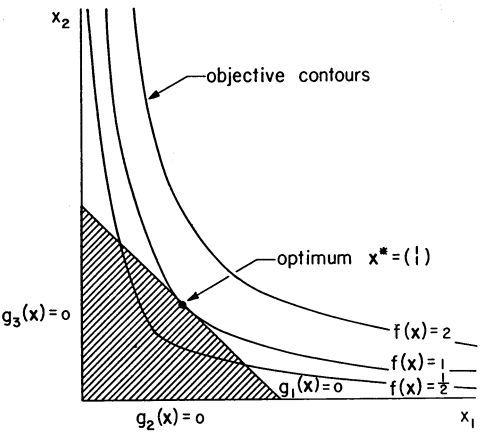


Figure 1 Graphical Solution of Example 1.

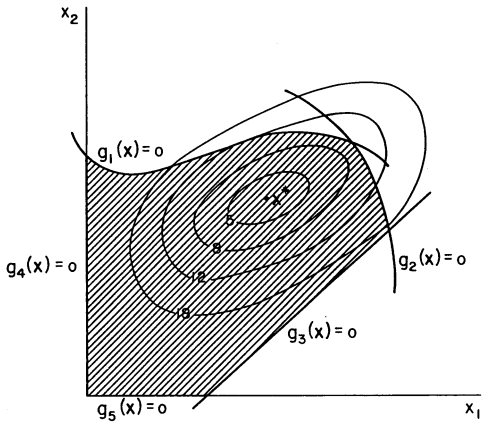


Figure 2 Example of a Nonlinear Programming Solution—No Active Constraints.

case, the constraints have no influence on the optimal solution, that is, the unconstrained optimum is also the constrained optimum.

This is just one of the difficulties encountered in nonlinear optimization. In contrast to linear programming, no single method is suitable for all types of nonlinear programs. And generally, it is only practical to obtain a *local minimum* that is not guaranteed to be a *global minimum*. (See Figure 3.)

Generally speaking, optimization methods can be divided into direct and indirect methods. Classical optimization methods are frequently referred to as *indirect methods* since these methods rely on analytic techniques for determining the optimal solution. This indirect approach does not involve a direct search for a particular solution but rather specifies a set of general conditions that must be satisfied by all solutions to the problem. As such, these methods do not lend themselves to computer implementations; however, they form the foundation for many of the computer-oriented direct methods. *Direct methods* of optimization seek to find a particular solution to a specified problem in a direct, iterative manner. The iterative nature of these procedures allows for effective computer implementations. Since real-world problems may involve many variables and constraints, the major challenge of nonlinear programming has been the development of efficient computational and algorithmic techniques. Although many algorithms have been developed, only a relative few have enjoyed continued success in real-world applications.

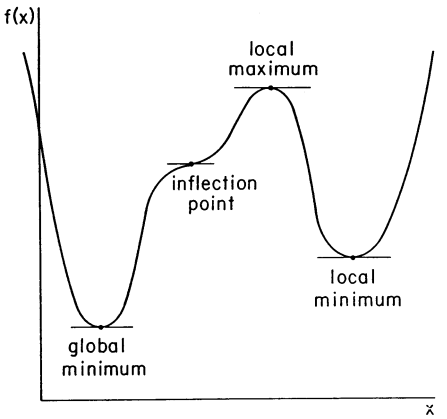


Figure 3 Local and Global Extrema.

This chapter outlines the fundamentals of nonlinear optimization. A review of some basic concepts and convexity is followed by a section addressing unconstrained optimization. This section presents classical optimization results as well as several direct search algorithms for solving univariate and multivariate problems. Next, constrained optimization problems are addressed. A review of Lagrange multipliers and the Karush–Kuhn–Tucker conditions precedes the presentation of several algorithms for solving different classes of nonlinear programming problems. There are a multitude of direct search algorithms in the literature, and the author has selected several representative methods to review. This is by no means an exhaustive review. It is simply an attempt to expose the reader to a cross-section of algorithmic methods. Finally, a listing of some nonlinear programming codes and some online resources is provided.

2. CONVEXITY

As indicated previously, nonlinear programming algorithms can generally only find a local optimal solution. Under these conditions, it becomes very important to know when a local minimum is actually guaranteed to be a global minimum over a given feasible region. One set of conditions that ensures such an outcome is that the feasible region is a convex set and the objective function is a convex function. Such a problem is called a *convex program*.

A *convex set* satisfies the following property: if $\mathbf{x}_1, \mathbf{x}_2 \in S$, then $\bar{\mathbf{x}} = \lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2 \in S$ for all $\lambda \in [0, 1]$. $\bar{\mathbf{x}}$ is referred to as a *convex combination* of $\mathbf{x}_1$ and $\mathbf{x}_2$. Geometrically, the above property simply means that if $\mathbf{x}_1, \mathbf{x}_2 \in S$, then the line segment joining $\mathbf{x}_1$ and $\mathbf{x}_2$ is also contained in S. An *extreme point* of a convex set S is any point in S that cannot be written as a strict convex combination (i.e. $\lambda \in (0, 1)$) of two distinct points in S. Figure 4 illustrates a convex and a nonconvex set. A set is *closed* if it contains all of its boundary points.

A function $f : S \rightarrow E_1$ defined on a convex set $S \subset E_n$ is a *convex function* if $f(\bar{\mathbf{x}}) \leq \lambda f(\mathbf{x}_1) + (1 - \lambda)f(\mathbf{x}_2)$ for all $\mathbf{x}_1, \mathbf{x}_2 \in S$ and $\lambda \in [0, 1]$. This property is illustrated geometrically in Figure 5(*a*) and means that the line segment joining the points $(\mathbf{x}_1, f(\mathbf{x}_1))$ and $(\mathbf{x}_2, f(\mathbf{x}_2))$ lies above the graph of $f(\mathbf{x})$. Also, f is a *strictly convex function* if $f(\bar{\mathbf{x}}) < \lambda f(\mathbf{x}_1) + (1 - \lambda)f(\mathbf{x}_2)$ for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbf{S}$, $\mathbf{x}_1 \neq \mathbf{x}_2$, and $\lambda \in [0, 1]$. Similarly, $g : S \rightarrow E_1$ is (*strictly*) *concave* if $-g$ is (strictly) convex.

As a special case, if the feasible region of a linear program with nonnegative variables is nonempty, the feasible region is always a convex polyhedral set with a finite number of extreme points. If an optimal solution exists, an optimal extreme point among this finite set can be found quite efficiently using either the simplex algorithm, which was developed by Dantzig in 1947, or, more recently, by Karmarkar's interior point algorithm (Karmarkar 1984). An analogous approach is not possible with general nonlinear programs, since the feasible region cannot readily be reduced to a finite set of candidate solutions and a global optimal solution cannot necessarily be found by relying only on local information, as in the simplex method. The effects of convexity on the outcome of different mathematical programming problems can be summarized as follows.

1. *Minimizing an unconstrained convex objective function:* In this case, any local minimum will also be a global minimum. Furthermore, if the objective function is differentiable, any stationary point (i.e., a point at which all the first-order derivatives vanish) will be a global minimum.
2. *Maximizing an unconstrained concave objective function:* Any stationary point will be a global maximum, and any local maximum will also be a global maximum.
3. *Minimizing a convex objective function over a closed convex feasible region:* Again, any local minimum will be a global minimum. As mentioned earlier, this class of problems is labeled convex programming problems, and a global minimum, if one exists, may occur at either an interior point or a boundary point of the feasible region.

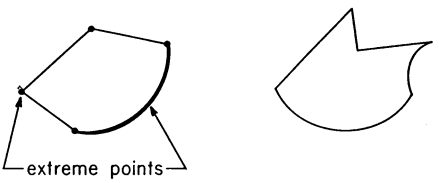


Figure 4 Examples of (*a*) Convex Set and (*b*) Nonconvex Set.

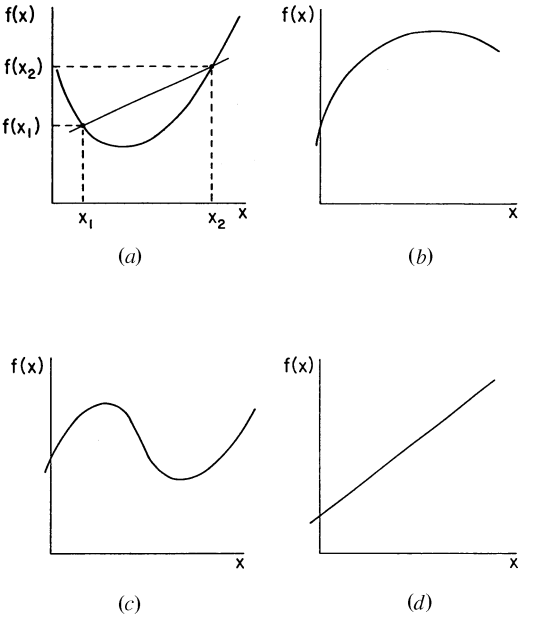


Figure 5 Examples of (*a*) Convex function, (*b*) Concave function, (*c*) Neither convex nor concave, (*d*) Both convex and concave.

4. *Maximizing a convex objective function over a closed bounded convex feasible region:* In this case, a maximum, if one exists, will be found at an extreme point of the feasible region. However, a local maximum need not be a global maximum.
5. *Maximizing a concave objective function over a closed convex feasible region:* This case is similar to case 3. That is, a local maximum is a global maximum.
6. *Minimizing a concave objective function over a closed bounded convex feasible region:* This case is analogous to case 4 and does not guarantee that a local minimum is a global minimum. However, a global minimum, if one exists, will occur at an extreme point of the feasible region.
7. *Maximizing or minimizing an objective function over a nonconvex set:* This is a difficult case that may produce a local minimum or maximum that is not also a global solution. Figure 6 illustrates that this can occur even in the simple case when the objective function is linear. Observe that extreme point **B** is a local minimum since all feasible points near **B** have an objective value that is greater than that at **B**.

In general, the task of proving that an arbitrary subset of E_n is convex can be quite difficult. However, the feasible region of problem (P) will be a convex set if each function $g_i(\mathbf{x})$ is a convex function and each function $h_j(\mathbf{x})$ is a linear function. Actually, these conditions on the functions $g_i(\mathbf{x})$, $h_j\mathbf{x})$ can be relaxed somewhat using the concepts of generalized convexity. A nice summary of generalized convexity is provided by Avriel (1976).

Verifying the convexity of an arbitrary function $f : E_n \rightarrow E_1$ can also be a difficult task. In the case when f is twice differentiable, however, the concept of quadratic form is useful for investigating the convexity or concavity of f.

A *quadratic form* $q(\mathbf{x})$ is defined by

$$q(\mathbf{x}) = \mathbf{x}^t\mathbf{A}\mathbf{x} \tag{8}$$

where $\mathbf{x} \in E_n$ and $\mathbf{A}$ is an $n \times n$ real symmetric matrix. The matrix $\mathbf{A}$ is said to be:

1. *Positive definite* if and only if $\mathbf{x}^t\mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$
2. *Negative definite* if $\mathbf{x}^t\mathbf{A}\mathbf{x} < 0$ and only if for all $\mathbf{x} \neq \mathbf{0}$

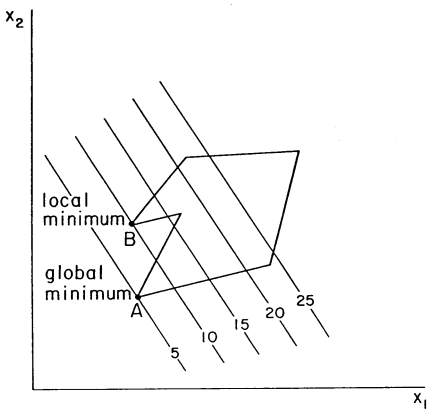


Figure 6 Minimizing a Linear Objective over a Nonconvex Set.

3. *Positive semidefinite* if and only if $\mathbf{x}^t\mathbf{A}\mathbf{x} \geq 0$ for all $\mathbf{x}$
4. *Negative semidefinite* if and only if $\mathbf{x}^t\mathbf{A}\mathbf{x} \leq 0$ for all $\mathbf{x}$
5. *Indefinite* if $\mathbf{x}^t\mathbf{A}\mathbf{x} > 0$ for some $\mathbf{x}$ and $\mathbf{x}^t\mathbf{A}\mathbf{x} < 0$ for some $\mathbf{x}$

Example 2. Consider the 2×2 symmetric matrix $\mathbf{A} = \begin{pmatrix} 1 & -3 \\ -3 & 11 \end{pmatrix}$. $\mathbf{A}$ is positive definite since

$$q = \mathbf{x}^t\mathbf{A}\mathbf{x} = (x_1, x_2)\begin{pmatrix} 1 & -3 \\ -3 & 11 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_2^2 - 6x_1x_2 + 11x_2^2 = (x_1 - 3x_2)^2 + 2x_2^2 > 0 \text{ for all } \mathbf{x} \neq \mathbf{0}$$

Positive and negative definiteness can also be checked using the leading principal minor test, although semidefiniteness cannot be verified in this manner. Using standard matrix notation, let

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \tag{9}$$

Define

$$A_1 = a_{11}, A_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$$

with $A_3, \ldots, A_n$ defined similarly. Note that $A_n = |\mathbf{A}|$, that is, A_n is the determinant of $\mathbf{A}$. A_i is called the ith *leading principal minor* of $\mathbf{A}$. The results of the *leading principal minor test* can be summarized as follows:

1. If $A_i > 0$ for all $i = 1, \ldots, n$, then $\mathbf{A}$ is positive definite.
2. If A_i, $i = 1, \ldots, n$, alternate in sign with $A_1 < 0$, then $\mathbf{A}$ is negative definite.
3. If 1 and 2 are not satisfied and $A_i \neq 0$ for all $i = 1, \ldots, n$, then $\mathbf{A}$ is indefinite.
4. If $A_i = 0$ for some i, then the test fails.

In Example 2, $A_1 = 1 > 0$ and $A_2 = 2 > 0$, therefore $\mathbf{A}$ is positive definite. A more comprehensive polynomial-time algorithm for checking both definiteness and semidefiniteness is presented in Bazaraa et al. (1994, pp. 96–97). The algorithm uses Gauss–Jordan reduction and works toward systematically reducing the matrix to upper triangular form until a conclusion is reached. Definiteness and semidefiniteness can also be determined by examining the eigenvalues of the matrix.

2.1. Convexity and the Hessian Matrix

Let $f : E_n \rightarrow E_1$ be a twice differentiable function and define the *Hessian matrix,* $\mathbf{H}(\mathbf{x})$, to be the matrix of second partial derivatives, $f_{ij}(\mathbf{x}) = \partial^2 f(\mathbf{x}) / \partial x_i \partial x_j$.

$$\mathbf{H}(\mathbf{x}) = \begin{pmatrix} f_{11}(\mathbf{x}) & f_{12}(\mathbf{x}) & \dots & f_{1n}(\mathbf{x}) \\ f_{21}(\mathbf{x}) & f_{22}(\mathbf{x}) & \dots & f_{2n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ f_{n1(}\mathbf{x}) & f_{n2}(\mathbf{x}) & \dots & f_{nn}(\mathbf{x}) \end{pmatrix} \tag{10}$$

Let $\bar{\mathbf{x}} \in E_n$. Then by examining the nature of the quadratic form of $\mathbf{H}(\mathbf{x})$, it is possible to determine local characteristics of the function f.

1. If $\mathbf{H}(\bar{\mathbf{x}})$ is positive definite, then f is locally convex at $\bar{\mathbf{x}}$.
2. If $\mathbf{H}(\bar{\mathbf{x}})$ is negative definite, then f is locally concave at $\bar{\mathbf{x}}$.
3. If $\mathbf{H}(\mathbf{x})$ is positive semidefinite for all $\mathbf{x}$, then f is a convex function.
4. If $\mathbf{H}(\mathbf{x})$ is negative semidefinite for all $\mathbf{x}$, then f is a concave function.

If f is a univariate function, that is, $f : E_1 \rightarrow E_1$, then the Hessian matrix simply reduces to the scalar, $f''(x)$ and the above conditions simplify to the following:

1. If $f''(\bar{x}) > 0$, then f is locally convex at $\bar{x}$.
2. If $f''(\bar{x}) < 0$, then f is locally concave at $\bar{x}$.
3. If $f''(x) \geq 0$ for all x, then f is a convex function.
4. If $f''(x) \leq 0$ for all x, then f is a concave function.

These conditions are used in the following section to develop optimality conditions in unconstrained optimization.

3. UNCONSTRAINED OPTIMIZATION

An unconstrained mathematical programming problem is a problem of the form: minimize $f(\mathbf{x})$ where $f : E_n \rightarrow E_1$. This is the type of problem that is typically addressed in a first course in calculus, and this class of problems has many applications, including maximum-likelihood or least-squares estimation problems in statistics. The classical fundamentals and algorithmic techniques used to solve these problems are often used as a basis for constructing efficient procedures for solving more general problems. This section presents some fundamental classical results, as well as algorithmic strategies for finding the solution of unconstrained optimization problems.

3.1. Classical Unconstrained Results

Classical methods of optimization are based on differential calculus, and it is generally assumed that the function to be optimized is continuous and differentiable (smooth). For a function of one variable, $f : E_1 \rightarrow E_1$, a *necessary condition* for a local extremum (either a local maximum or local minimum) to occur at a point $x^* \in E_1$ is that the first derivative vanishes at x^*, that is,

$$f'(x^*) = 0 \tag{11}$$

However, a local extremum does not necessarily occur at every point that satisfies (11); that is, (11) is not a *sufficient condition* for optimality. In practice, necessary conditions are used to identify *stationary points,* which are candidate extrema, whereas sufficient conditions are used to classify the stationary points as local maxima, local minima, or saddle points (inflection points in E_2). Once all local extrema are found, the global extrema can be found by selecting the absolute maximum or minimum. The necessary and sufficient conditions for determining and classifying the stationary points of a function of one variable are summarized in Table 1. These conditions are easily derived using a Taylor series expansion.

As indicated in the previous section, the sufficiency conditions in Table 1 are essentially examining the local properties of the function f. For example, $f''(\bar{x}) > 0$ indicates that the function is convex at the point $\bar{x}$, and thus, if $f'(\bar{x}) = 0$, a local minimum occurs at $\bar{x}$. Similarly, if $f''(\bar{x}) < 0$, then f is concave at $\bar{x}$.

To illustrate the use of the results presented in Table 1 consider the following simple problem.

Example 3. Find the extrema of the function $f(x) = x^3 (3x - 8) + 20$. First, the stationary points are identified by finding the roots of $f'(x) = 12x^2 (x - 2) = 0$. Clearly $x = 0, 2$ are the only stationary points. Next, these candidate points are classified by examining higher-order derivatives. In particular,

TABLE 1 Necessary and Sufficient Conditions for Local Extrema of a Univariate Function

	Necessary Condition	Sufficient Condition
Local minimum at $\bar{x}$	$f'(\bar{x}) = 0$	$f^{(i)}(\bar{x}) = 0$, $i = 1, \ldots, m-1$, $f^{(m)}(\bar{x}) > 0$ and m is even
Local maximum at $\bar{x}$	$f'(\bar{x}) = 0$	$f^{(i)}(\bar{x}) = 0$, $i = 1, \ldots, m-1$ $f^{(m)}(\bar{x}) < 0$ and m is even
Inflection point at $\bar{x}$	$f'(\bar{x}) = 0$	$f^{(i)}(\bar{x}) = 0$, $i = 1, \ldots, m-1$ $f^{(m)}(\bar{x}) \neq 0$ and m is odd

for this example, it is necessary to determine $f''(x) = 36x^2 - 48x$ and $f'''(x) = 72x - 48$. A local minimum occurs at $x = 2$ since $f''(2) = 48 > 0$ and $m = 2$ is even. Also, $f''(0) = 0$ and $f'''(0) = -48$. Therefore, an inflection point occurs at $x = 0$ since $f'''(0) < 0$ and $m = 3$ is odd. In addition, upon graphing, it can be seen that the local minimum at $x = 2$ is also a global minimum, even though the function f is not convex.

For a function of n variables, $f : E_n \rightarrow E_1$, a necessary condition for an extremum to occur at a point $\mathbf{x}^* \in E_n$ is that the gradient vector, $\nabla f(\mathbf{x})$, the vector of first partial derivatives, vanishes at $\mathbf{x}^*$. Thus, in this case, the task of finding the stationary points is more difficult, in that it is necessary, in general, to solve a system of n simultaneous nonlinear equations in n unknowns,

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 0 \quad \text{for } i = 1, \ldots, n \tag{12}$$

Sufficiency conditions are determined by examining the Hessian matrix $\mathbf{H}(\mathbf{x})$, the matrix of second partial derivatives. Based on the discussion in Section 2, Table 2 gives the necessary and sufficiency conditions for identifying and classifying stationary points in the multivariate case. In the event that the sufficiency conditions are not satisfied, higher-order derivative information must be used.

Example 4. Find the extrema of the function $f(\mathbf{x}) = x_1^3 - 6x_1x_2 + 24x_1 + x_2^2$. First, obtain first and second order derivative information.

$$\nabla f(\mathbf{x}) = \begin{pmatrix} 3x_1^2 - 6x_2 + 24 \\ -6x_1 + 2x_2 \end{pmatrix} \tag{13}$$

$$\mathbf{H}(\mathbf{x}) = \begin{pmatrix} 6x_1 & -6 \\ -6 & 2 \end{pmatrix} \tag{14}$$

Setting $\nabla f(\mathbf{x}) = 0$, the stationary points are $\mathbf{x}_1 = (2, 6)^t$ and $\mathbf{x}_2 = (4, 12)^t$. From Table 2, $\mathbf{x}_1$ is a saddle point since $H_1(\mathbf{x}_1) = 12$ and $H_2(\mathbf{x}_1) = -12$, that is, $\mathbf{H}(\mathbf{x}_1)$ is indefinite. Similarly, $H_1(\mathbf{x}_2) = 24$ and $H_2(\mathbf{x}_2) = 12$ implies that $\mathbf{H}(\mathbf{x}_2)$ is positive definite and thus, a local minimum occurs at $\mathbf{x}_2$.

3.2. Line Search Techniques

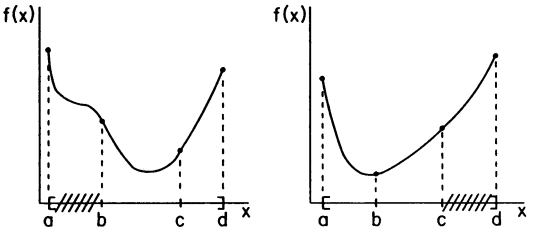
There are several direct search techniques for minimizing a function of one variable. The methods generally start from an initial estimate and sequentially move toward the minimum. Univariate or line search techniques play a major role in solving subproblems in more complex direct search algorithms.

3.2.1. Golden Section Method

An example of a line search technique that does not use derivatives is the golden section method, which seeks to find the minimum of a function $f(x)$ on an interval $[a, d]$. The interval $[a, d]$ is called

TABLE 2 Necessary and Sufficient Conditions for Local Extrema of a Multivariate Function

	Necessary Condition	Sufficient Condition
Local minimum at $\bar{\mathbf{x}}$	$\nabla f(\bar{\mathbf{x}}) = \mathbf{0}$	$\mathbf{H}(\bar{\mathbf{x}})$ is positive definite
Local maximum at $\bar{\mathbf{x}}$	$\nabla f(\bar{\mathbf{x}}) = \mathbf{0}$	$\mathbf{H}(\bar{\mathbf{x}})$ is negative definite
Saddle point at $\bar{\mathbf{x}}$	$\nabla f(\bar{\mathbf{x}}) = \mathbf{0}$	$\mathbf{H}(\bar{\mathbf{x}})$ is indefinite



(*a*) (*b*)

Figure 7 Narrowing the Interval of Uncertainty.

the *interval of uncertainty,* and the basic strategy of the method is to successively decrease the length of the interval, assuming that the function is unimodal, that is, the function has only one minimum on the interval $[a, d]$.

Let $b, c \in [a, d]$ such that $a < b < c < d$ and compute $f(b)$ and $f(c)$. Since f is unimodal on $[a, d]$, it is possible to narrow the interval of uncertainty by comparing $f(b)$ and $f(c)$. That is, if $f(b) > f(c)$, then the true minimum cannot lie on the interval $[a, b]$ [Figure 7(*a*)], and if $f(b) < f(c)$, the minimum cannot lie on the interval $[c, d]$ [Figure 7(*b*)]. Thus, in either case, the length of the interval of uncertainty is reduced and the entire process can be repeated until some desired accuracy is achieved.

Let $a_k < b_k < c_k < d_k$, where $[a_k, d_k]$ is the interval of uncertainty at iteration k and assume that $[a_1, d_1] = [a, d]$. Since functional evaluations are the most expensive step in the process, the golden section method reduces the amount of overall work by intelligently choosing symmetric points b_k and c_k so that they can be reused on successive iterations, as illustrated in Figure 8. This is achieved by using the relationships $b_k = \lambda a_k + (1 - \lambda)d_k$ and $c_k = (1 - \lambda)a_k + \lambda d_k$, where $\lambda = 0.618$. Observe that b_k and c_k are simply expressed as convex combinations of a_k and d_k. A summary of the *golden section algorithm* follows:

1. Choose a tolerance $\varepsilon > 0$ for the length of the final interval of uncertainty. Choose an initial interval of uncertainty $[a_1, d_1]$ and set the iteration counter $k = 1$. Compute $b_1 = \lambda a_1 + (1 - \lambda)d_1$ and $c_1 = (1 - \lambda)a_1 + \lambda d_1$.
2. If $d_k - a_k < \varepsilon$, stop; the interval $[a_k, d_k]$ contains the minimum. Otherwise, continue with step 3.
3. If $f(b_k) > f(c_k)$, go to step 4. Otherwise go to step 5.
4. Let $a_{k+1} = b_k$, $b_{k+1} = c_k$, $d_{k+1} = d_k$ [see Figure 8(*a*)]. Compute $c_{k+1} = (1 - \lambda)a_{k+1} + \lambda d_{k+1}$. Replace k by $k + 1$ and go to step 2.
5. Let $a_{k+1} = a_k$, $c_{k+1} = b_k$, $d_{k+1} = c_k$. [see Figure 8(*b*)]. Compute $b_{k+1} = \lambda a_{k+1} + (1 - \lambda)d_{k+1}$. Replace k by $k + 1$ and go to step 2.

Other line search methods that involve only function evaluations, that is, no derivative calculations, are the dichotomous search, the Fibonacci search (Kiefer 1957), and the quadratic fit line search. The Fibonacci search is the most efficient derivative-free line search technique in the sense that it requires the fewest function evaluations to attain a prescribed degree of accuracy. The quadratic fit method

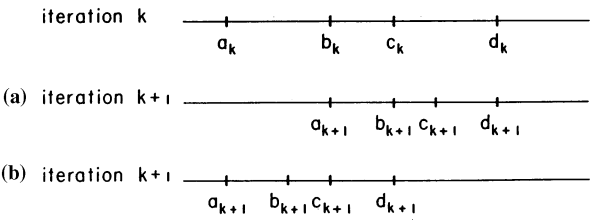


Figure 8 Successive Iterations of the Golden Section Method.

fits a quadratic function to three points on the function and then finds the unique minimizing point of the resulting quadratic function. A new interval of uncertainty is determined and the process is repeated until convergence is achieved (see, e.g., Bazaraa et al. 1994, pp. 279–281). Examples of line search techniques that utilize derivative information are the bisection search method and Newton's method. The more general multivariate version of Newton's algorithm will be discussed in the next section. In practice, a line search is typically used to find the step length during an iterative step of a more general algorithm. As such, it may be impractical to try to find the exact minimum point and it may be appropriate to apply an inexact method such as Armijos' rule (Armijo 1966) or simply to terminate the line search procedure before it has converged.

3.3. Multidimensional Search Techniques

This section is a natural extension of the previous section and addresses the problem of minimizing a function of several variables, that is, minimize $f(\mathbf{x})$ where $f : E_n \rightarrow E_1$. The general process behind multidimensional search techniques may be expressed as follows: Given a current point $\mathbf{x}_k$, determine a direction $\mathbf{d}_k$ and a step size α_k to yield a new point,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \tag{15}$$

The step size α_k is determined by either solving the line search problem in the variable α,

$$\text{Minimize } f(\mathbf{x}_k + \alpha \mathbf{d}_k) \tag{16}$$

where typically $\alpha \in E_1$, $\alpha \in [0, \infty)$, or $\alpha \in [a, b]$, or by taking prescribed discrete steps along the search directions. The strategy for choosing the search directions and the step sizes determines the different methods.

3.3.1. Multidimensional Search Techniques without Using Derivatives

The *cyclic coordinate search* method successively uses search directions that are parallel to the coordinate axes along with line search problems to determine the step sizes. This method is conceptually simple, but the sequence of iterates generated by this method tend to zigzag if the optimal solution lies in a valley.

To help overcome this problem, the method of Hooke and Jeeves (1961) involves both exploratory moves and pattern moves (acceleration moves) with discrete steps along the search directions. The discrete steps eliminate the need for a line search. In the exploratory move phase, starting at a point $\mathbf{x}_k$, a modified cyclic coordinate search is performed in which, if possible, the objective function is reduced once along each of the coordinate directions using a prescribed discrete step length. This leads to a new point $\mathbf{x}_{k+1}$ and establishes a direction of improvement. A pattern move is then performed in the direction $\mathbf{x}_{k+1} - \mathbf{x}_k$, leading to an intermediate point $\mathbf{y}$. Now, starting from $\mathbf{y}$, another exploratory move yields the point $\mathbf{x}_{k+2}$. If $f(\mathbf{x}_{k+2} < f(\mathbf{x}_{k+1})$, then an improvement has been found and the process is continued with a pattern move in the direction $\mathbf{x}_{k+2} - \mathbf{x}_{k+1}$. Otherwise, $\mathbf{x}_{k+1}$ is considered the new initial point and the process is begun anew. In the instance when no improving exploratory step can be made, the discrete step size is reduced and the process is repeated. When the step size becomes smaller than some prescribed tolerance, the search terminates. A graphical interpretation of the algorithm is presented in Figure 9.

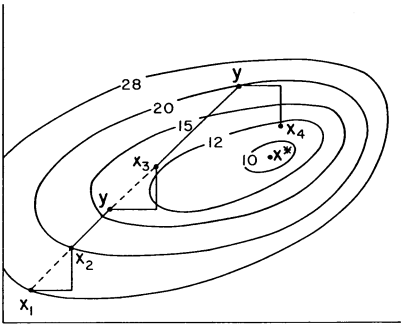


Figure 9 The Method of Hooke and Jeeves.

Other methods of multidimensional search without using derivatives include Rosenbrock's method (1960) and the simplex method of Spendley et al. (1962), which was later modified by Nelder and Meade (1974). Although it has the same name, this simplex method is not the same algorithm as that used for linear programming; it is a polytope algorithm that requires only functional evaluations and requires no smoothness assumptions.

3.3.2. Multidimensional Search Techniques Using Derivatives

In this section, optimization techniques are discussed that use derivative information in determining the search directions. As in the preceding discussion, a basic step in the algorithmic process consists of choosing a direction $\mathbf{d}_k$ and a step length α_k to arrive at a new point,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \tag{17}$$

Recall from calculus that the gradient, $\nabla f(\mathbf{x})$, of a differentiable function $f : E_n \rightarrow E_1$ provides local information concerning the rate of change of the function. In fact, given a point $\bar{\mathbf{x}}$, the gradient of f evaluated at $\bar{\mathbf{x}}$, $\nabla f(\bar{\mathbf{x}})$, is the direction of steepest ascent at $\bar{\mathbf{x}}$ and $-\nabla f(\bar{\mathbf{x}})$ is the direction of steepest descent at $\bar{\mathbf{x}}$. This fundamental result leads to the following algorithm.

3.3.2.1. The Method of Steepest Descent This is one of the most basic procedures for minimizing an unconstrained differentiable function, and the process is defined by simply specifying $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$ in Eq. (17) to get

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) \tag{18}$$

where α_k is determined by solving the line search problem

$$\underset{\alpha \geq 0}{\text{Minimize}}\ f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)) \tag{19}$$

The path generated by the algorithm for an example problem is illustrated in Figure 10. Note that successive directions are orthogonal and at each point $\mathbf{x}_k$, $\nabla f(\mathbf{x}_k)$ is normal to the objective contour at $\mathbf{x}_k$. A typical stopping rule is when the Euclidean norm of $\nabla f(\mathbf{x}_k)$, $(\|\nabla f(\mathbf{x}_k)\|)$, is less than some prescribed small positive number. The performance of the method of steepest descent is quite good during early iterations, but the convergence tends to slow excessively during later iterations and exhibits zigzagging tendencies near stationary points. Thus, it is not well used in practice.

3.3.2.2. Newton's Method The classical Newton's method is a technique that instead of specifying a step length at each iteration uses the inverse of the Hessian matrix, $\mathbf{H}(\mathbf{x})^{-1}$, to deflect the direction of steepest descent. The method assumes that $f(\mathbf{x})$ may be approximated locally by a second order Taylor approximation and is derived quite easily by determining the minimum point of this quadratic approximation. Assuming that $\mathbf{H}(\mathbf{x}_k)$ is nonsingular, then the algorithmic process is defined by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k) \tag{20}$$

The classical method does not require a line search at each iteration, but it does require second order derivative information and a matrix inversion, or equivalently, the solution of a system of linear

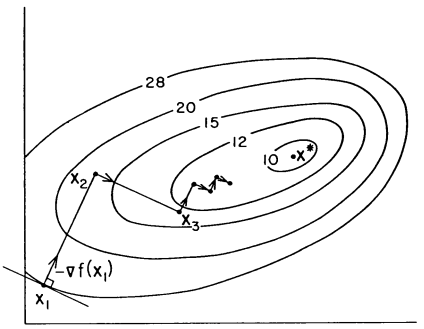


Figure 10 The Steepest Descent Method.

equations. In general, the performance of Newton's method is quite erratic and the sequence of points generated may not converge. Thus, it is not well suited for general use. However, Newton's method performs quite well if the initial point is sufficiently close to the optimal solution. This is because near an optimal solution, the local contours of the approximating quadratic function are usually a good approximation of those of the function.

Another interesting feature is that the method is not a true descent procedure, since the objective function is not guaranteed to decrease at each iteration. However, the direction, $-\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$, is guaranteed to be a descent direction if $\mathbf{H}(\mathbf{x}_k)$ is positive definite. Thus, adding a line search to control the step size means the objective will improve at each iteration provided that $\mathbf{H}(\mathbf{x}_k)$ is positive definite. In the case when $\mathbf{H}(\mathbf{x}_k)$ is indefinite, some other corrective action must be taken. One alternative is to give the Newton search direction a bias towards the steepest descent direction as suggested by Levenberg (1944) and Marquardt (1963). Another method, suggested by Fiacco and McCormick (1990, pp. 167–169), involves computing a negative curvature descent direction when $\mathbf{H}(\mathbf{x}_k)$ is indefinite. For further discussion of these methods, see, for example, McCormick (1983).

Example 5. Use classical Newton's method to minimize $f(\mathbf{x}) = (x_1^2 - x_2)^2 + (1 - x_1)^2$. Clearly, since $f(\mathbf{x}) \geq 0$ for all $\mathbf{x}$, a unique minimum occurs at $\mathbf{x}^* = (1, 1)^t$. First, compute the gradient vector, $\nabla f(\mathbf{x})$, and the Hessian matrix, $\mathbf{H}(\mathbf{x})$.

$$\nabla f(\mathbf{x}) = \begin{pmatrix} 4x_1^3 - 4x_1x_2 + 2x_1 - 2 \\ -2x_1^2 + 2x_2 \end{pmatrix} \tag{21}$$

$$\mathbf{H}(\mathbf{x}) = \begin{pmatrix} 12x_1^2 - 4x_2 + 2 & -4x_1 \\ -4x_1 & 2 \end{pmatrix} \tag{22}$$

Choosing $\mathbf{x}_1 = (-1, 1)^t$, then from (20),

$$\mathbf{x}_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} - \begin{pmatrix} 10 & 4 \\ 4 & 2 \end{pmatrix}^{-1} \begin{pmatrix} -4 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ -3 \end{pmatrix} \tag{23}$$

$$\mathbf{x}_3 = \begin{pmatrix} 1 \\ -3 \end{pmatrix} - \begin{pmatrix} 26 & -4 \\ -4 & 2 \end{pmatrix}^{-1} \begin{pmatrix} 16 \\ -8 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{24}$$

Observe that $f(\mathbf{x}_1) = 4$, $f(\mathbf{x}_2) = 16$, and $f(\mathbf{x}_3) = 0$. That is, even though the minimum was found in two iterations, the objective function did not improve from iteration 1 to iteration 2.

3.3.2.3. Quasi-Newton Methods In some sense, quasi-Newton methods are an attempt to combine the best features of the steepest descent method with those of Newton's method. Recall that the steepest descent method performs well during early iterations and always decreases the value of the function, whereas Newton's method performs well near the optimum but requires second order derivative information. Quasi-Newton methods are designed to start like the steepest descent method and finish like Newton's method while using only first order derivative information. The basic idea was originally proposed by Davidon (1959) and subsequently developed by Fletcher and Powell (1963). An additional feature of quasi-Newton methods is that the minimum of a convex quadratic function $f : E_n \rightarrow E_1$ can be found in at most n iterations if exact line searches are used. The basic step of the algorithm is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{D}_k \nabla f(\mathbf{x}_k) \tag{25}$$

where $\mathbf{D}_k$ is a deflection matrix requiring only first order derivative information and α_k is determined using a line search. The method of Davidon–Fletcher–Powell can be summarized as follows:

1. Choose a small positive number $\varepsilon > 0$ to test convergence and choose an initial point $\mathbf{x}_1$. Let the initial deflection matrix $\mathbf{D}_1 = \mathbf{I}$ and set the iteration counter $k = 1$.
2. Compute $\nabla f(\mathbf{x}_k)$. If $\|\nabla f(\mathbf{x}_k)\| < \varepsilon$, stop with $\mathbf{x}_k$ as the optimal solution. Otherwise, set $\mathbf{d}_k = -\mathbf{D}_k \nabla f(\mathbf{x}_k)$ and continue with step 3.
3. Let $\alpha = \alpha_k$ be a solution to the univariate problem, $\underset{\alpha \geq 0}{\text{minimize}}\ f(\mathbf{x}_k + \alpha \mathbf{d}_k)$.
4. Compute $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$.
5. Find the updated deflection matrix $\mathbf{D}_{k+1}$ using

$$\mathbf{D}_{k+1} = \mathbf{D}_k + \frac{\mathbf{u}_k \mathbf{u}_k^t}{\mathbf{u}_k^t \mathbf{v}_k} - \frac{\mathbf{D}_k \mathbf{v}_k \mathbf{v}_k^t \mathbf{D}_k}{\mathbf{v}_k^t \mathbf{D}_k \mathbf{v}_k} \tag{26}$$

where $\mathbf{u}_k = \alpha_k \mathbf{d}_k$ and $\mathbf{v}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$. Replace k by $k + 1$ and go to step 2.

Since $\mathbf{D}_1 = \mathbf{I}$, the initial iteration is a steepest descent step. Also, if f is a convex quadratic function, it can be shown that at termination $\mathbf{D}_k$ is the inverse of the Hessian matrix.

There are several other methods for updating $\mathbf{D}_k$ in step 5. Another important method is based on the BFGS formula, which was developed by Broyden (1970), Fletcher (1970), Goldfarb (1970), and Shanno (1970) and is given by

$$\mathbf{D}_{k+1} = \mathbf{D}_k + \left(1 + \frac{\mathbf{v}_k^t \mathbf{D}_k \mathbf{v}_k}{\mathbf{u}_k^t \mathbf{v}_k}\right) \frac{\mathbf{u}_k \mathbf{u}_k^t}{\mathbf{u}_k^t \mathbf{v}_k} - \frac{\mathbf{u}_k \mathbf{v}_k^t \mathbf{D}_k + \mathbf{D}_k \mathbf{v}_k \mathbf{u}_k^t}{\mathbf{u}_k^t \mathbf{v}_k} \tag{27}$$

The BFGS formula is generally preferred to (26) since computational results have shown that it requires considerably less effort, especially when inexact line searches are used. Quasi-Newton methods, also referred to as variable metric methods, are much more widely used than either the steepest descent or Newton's method. For additional details and computational comparisons, see Fletcher (1987, pp. 44–74).

Example 6. Find the minimum of $f(\mathbf{x}) = x_1^2/2 - x_1 x_2 + x_2^2 - x_1 - x_2$ using the method of Davidon–Fletcher–Powell. Choose an initial point $\mathbf{x}_1 = (0, 0)^t$, an initial deflection matrix $\mathbf{D}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, and determine the gradient vector,

$$\nabla f(\mathbf{x}) = \begin{pmatrix} x_1 - x_2 - 1 \\ -x_1 + 2x_2 - 1 \end{pmatrix} \tag{28}$$

Next, compute $\nabla f(\mathbf{x}_1) = (-1, -1)^t$, and as in step 3, solve the line search problem:

$$\underset{\alpha \geq 0}{\text{minimize}}\ f\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} - \alpha \begin{pmatrix} -1 \\ -1 \end{pmatrix}\right) = \alpha^2/2 - 2\alpha \tag{29}$$

to find $\alpha_1 = 2$. Therefore, from (25),

$$\mathbf{x}_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \tag{30}$$

Now compute the updated deflection matrix as in (26).

$$\mathbf{D}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{\begin{pmatrix} 2 \\ 2 \end{pmatrix}(2 \quad 2)}{(2 \quad 2)\begin{pmatrix} 2 \\ 2 \end{pmatrix}} - \frac{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 2 \end{pmatrix}(0 \quad 2)\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}{(0 \quad 2)\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 2 \end{pmatrix}} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \tag{31}$$

Then,

$$\mathbf{x}_3 = \begin{pmatrix} 2 \\ 2 \end{pmatrix} - 1 \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} - \begin{pmatrix} 3 \\ 2 \end{pmatrix} \tag{32}$$

Since $\nabla f(\mathbf{x}_3) = 0$, the algorithm terminates with $\mathbf{x}^* = (3, 2)^t$. Note that $\mathbf{D}_2$ is precisely the inverse of the Hessian matrix, $\mathbf{H}(\mathbf{x}) = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}$, of the convex quadratic objective function $f(\mathbf{x})$.

3.4. Conjugate Gradient Methods

The nonzero vectors $\mathbf{d}_1, \ldots, \mathbf{d}_k$ are said to be *conjugate* with respect to the positive definite matrix $\mathbf{H}$ if they are linearly independent and $\mathbf{d}_i^t \mathbf{H} \mathbf{d}_j = 0$ for $i \neq j$. A method that generates such directions when applied to a quadratic function with Hessian matrix $\mathbf{H}$ is called a *conjugate direction method*. These methods will locate the minimum of a quadratic function in a finite number of iterations, and they can also be applied iteratively to optimize nonquadratic functions.

The algorithm proposed by Fletcher and Reeves (1964), is probably the best-known example of a conjugate gradient algorithm. Starting at some initial point $\mathbf{x}_1$, the algorithm begins like the steepest descent method by searching in the direction $\mathbf{d}_1 = -\nabla f(\mathbf{x}_1)$ to determine $\mathbf{x}_2$. Subsequent directions are computed using the expression

$$\mathbf{d}_{k+1} = -\nabla f(\mathbf{x}_{k+1}) + \alpha_k \mathbf{d}_k \tag{33}$$

where

$$\alpha_k = \frac{\nabla f(\mathbf{x}_{k+1})^t \nabla f(\mathbf{x}_{k+1})}{\nabla f(\mathbf{x}_k)^t \nabla f(\mathbf{x}_k)} \tag{34}$$

Note from (33) that the method actually deflects the current direction of steepest descent by adding on a positive multiple of the direction vector used in the previous step. In fact, it is easily shown that $\mathbf{d}_{k+1}$ is essentially a convex combination of $-\nabla f(\mathbf{x}_{k+1})$ and $\mathbf{d}_k$.

Quasi-Newton methods are generally preferable to conjugate gradient methods because they have been shown to be more efficient. However, conjugate gradient methods have the advantage of requiring no matrix operations and having minimal storage requirements for computer implementation.

4. CONSTRAINED OPTIMIZATION

In this section, methods for addressing constrained optimization problems are discussed. First the classical method of Lagrange multipliers is presented, followed by the Karush–Kuhn–Tucker conditions. Both of these techniques are indirect methods, and general optimality conditions are presented. Then several algorithmic strategies are presented for addressing specific classes of nonlinear programming problems.

4.1. Lagrange Multipliers

The method of Lagrange multipliers is a classical indirect method for solving nonlinear programming problems in which the objective is to optimize a differentiable function subject to a set of equality constraints comprised of differentiable functions. The basic technique involves transforming the constrained nonlinear problem into an unconstrained nonlinear problem by forming what is called the Lagrangian function. Consider the mathematical programming problem with equality constraints,

$$\text{(NLPE) Minimize } f(\mathbf{x}) \tag{35}$$

$$\text{Subject to } h_j(\mathbf{x}) = 0 \text{ for } j = 1, \ldots, p \tag{36}$$

The *Lagrangian function,*

$$L(\mathbf{x}, \mathbf{v}) = f(\mathbf{x}) + \sum_{j=1}^{p} v_j h_j(\mathbf{x}) \tag{37}$$

is formed by introducing a vector of Lagrange multipliers, $\mathbf{v} = (v_1, \ldots, v_p)^t$. v_j is referred to as the Lagrange multiplier, dual multiplier, or dual variable for constraint $h_j(\mathbf{x}) = 0$. Since each feasible point $\mathbf{x}$ must satisfy $h_j(\mathbf{x}) = 0$ for all $j = 1, \ldots, p$, the addition of the term $\Sigma_{j=1}^{p} v_j h_j(\mathbf{x})$ to the objective function $f(\mathbf{x})$ does not change the value of the function. As in section 3, a necessary condition for a stationary point is that the partial derivatives with respect to x_i for $i = 1, \ldots, n$ and v_j for $j = 1, \ldots, p$, equal zero. That is,

$$\frac{\partial L(\mathbf{x}, \mathbf{v})}{\partial x_i} = \frac{\partial f(\mathbf{x})}{\partial x_i} + \sum_{j=1}^{p} v_j \frac{\partial h_j(\mathbf{x})}{\partial x_i} = 0 \quad \text{for } i = 1, \ldots, n \tag{38}$$

$$\frac{\partial L(\mathbf{x}, \mathbf{v})}{\partial v_j} = h_j(\mathbf{x}) = 0 \quad \text{for } j = 1, \ldots, p \tag{39}$$

A stationary point for a general Lagrangian function may or may not be a local extremum. If, as described in Section 2, suitable convexity conditions hold, then the method of Lagrange multipliers will yield a global minimum.

Example 7

$$\text{Minimize } f(\mathbf{x}) = x_1^2 + 2x_2^2 + 4x_1x_2 - 12x_1 - 15x_2 \tag{40}$$

$$\text{Subject to } h_1(\mathbf{x}) = 2x_1 - x_2 - 5 = 0 \tag{41}$$

First, form the Lagrangian function,

$$L(\mathbf{x}, \mathbf{v}) = x_1^2 + 2x_2^2 + 4x_1x_2 - 12x_1 - 15x_2 + v_1(2x_1 - x_2 - 5) \tag{42}$$

Then, using (38) and (39),

$$\frac{\partial L(\mathbf{x}, \mathbf{v})}{\partial x_1} = 2x_1 + 4x_2 - 12 + 2v_1 = 0 \tag{43}$$

$$\frac{\partial L(\mathbf{x}, \mathbf{v})}{\partial x_2} = 4x_1 + 4x_2 - 15 - v_1 = 0 \tag{44}$$

$$\frac{\partial L(\mathbf{x}, \mathbf{v})}{\partial v_1} = 2x_1 - x_2 - 5 = 0 \tag{45}$$

Solving the system of linear equations, (43) through (45), yields $x_1 = 3$, $x_2 = 1$, and $v_1 = 1$. Thus, the optimal solution is $\mathbf{x}^* = (3, 1)^t$.

4.2. Karush–Kuhn–Tucker Conditions

Consider the general nonlinear programming problem,

$$\text{(NLP) Minimize } f(\mathbf{x}) \tag{46}$$

$$\text{Subject to } g_i(\mathbf{x}) \leq 0 \text{ for } i = 1, \ldots, m \tag{47}$$

$$h_j(\mathbf{x}) = 0 \text{ for } j = 1, \ldots, p \tag{48}$$

where f, g_i, $h_j : E_n \rightarrow E_1$ are continuously differentiable functions. The *Karush–Kuhn–Tucker (*KKT*) conditions* are essentially an extension of the method of Lagrange multipliers to problems involving inequality constraints. The results were derived independently by Karush (1939), and Kuhn and Tucker (1951). Although these necessary conditions are not typically used to derive an optimal solution, they do provide insight into solution behavior and form the basis for many nonlinear programming algorithms. They can also be used to verify that a given solution is a candidate for an optimal solution.

4.2.1. KKT Necessary Conditions

If $\mathbf{x}^*$ is a local minimum of problem (NLP) and some constraint qualification holds (e.g. the vectors $\nabla g_i(\mathbf{x}^*)$ for all i such that $g_i(\mathbf{x}^*) = 0$, and $\nabla h_j(\mathbf{x}^*)$ for $j = 1, \ldots, p$, are linearly independent), then there exists vectors $\mathbf{u} = (u_1, \ldots, u_m)^t$ and $\mathbf{v} = (v_1, \ldots, v_p)^t$ such that

$$g_i(\mathbf{x}^*) \leq 0 \text{ for } i = 1, \ldots, m \tag{49}$$

$$h_j(\mathbf{x}^*) = 0 \text{ for } j = 1, \ldots, p \tag{50}$$

$$\frac{\partial f(\mathbf{x}^*)}{\partial x_k} + \sum_{i=1}^{m} u_i \frac{\partial g_i(\mathbf{x}^*)}{\partial x_k} + \sum_{j=1}^{p} v_j \frac{\partial h_j(\mathbf{x}^*)}{\partial x_k} = \text{ for } k = 1, \ldots, n \tag{51}$$

$$u_i \geq 0 \text{ for } i = 1, \ldots, m \tag{52}$$

$$u_i g_i(\mathbf{x}^*) = 0 \text{ for } i = 1, \ldots, m \tag{53}$$

Conditions (49) and (50) are called *primal feasibility,* conditions (51) and (52) are called *dual feasibility,* and condition (53) is called *complementary slackness*. To interpret these results geometrically, it is helpful to rewrite (51) in a more compact notation using gradients,

$$-\nabla f(\mathbf{x}^*) = \sum_{i=1}^{m} u_i \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^{p} v_j \nabla h_j(\mathbf{x}^*) \tag{54}$$

Given a point $\mathbf{x}^*$, condition (53) asserts that the dual variable u_i will be zero if $g_i(\mathbf{x}^*) < 0$ (i.e., the constraint $g_i(\mathbf{x}) \leq 0$ is nonbinding at $\mathbf{x}^*$). Thus, from (53) and (54), the KKT conditions are specifying that, at an optimal solution, $-\nabla f(\mathbf{x}^*)$ can be written as a linear combination of the gradients of the binding constraints (i.e., those constraints satisfied as equalities). In the case involving only constraints of the form $g_i(\mathbf{x}) \leq 0$, $-\nabla f(\mathbf{x}^*)$ must lie in the cone spanned by the gradients of the binding constraints since $u_i \geq 0$ for $i = 1, \ldots, m$. This geometric interpretation is illustrated in Figure 11 using the following example.

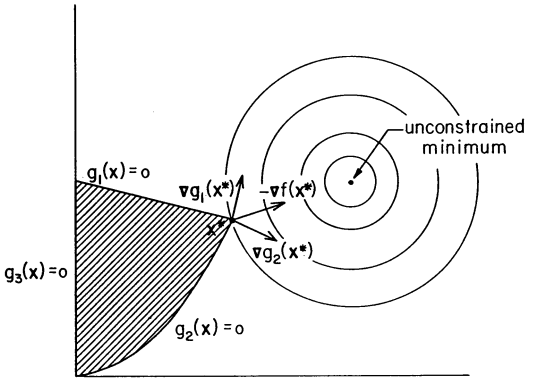


Figure 11 Graphical Solution of Example 8 Illustrating the KKT Conditions.

Example 8

$$\text{Minimize } f(\mathbf{x}) = (x_1 - 14)^2 + (x_2 - 12)^2 \tag{55}$$

$$\text{Subject to } g_1(\mathbf{x}) = x_1 + 4x_2 - 40 \leq 0 \tag{56}$$

$$g_2(\mathbf{x}) = x_1^2 - 8x_2 \leq 0 \tag{57}$$

$$g_3(\mathbf{x}) = -x_1 \leq 0 \tag{58}$$

The KKT conditions for this problem can be written as follows:

$$x_1 + 4x_2 - 40 \leq 0 \tag{59}$$

$$x_1^2 - 8x_2 \leq 0 \tag{60}$$

$$-x_1 \leq 0 \tag{61}$$

$$\begin{pmatrix} 2x_1 - 28 \\ 2x_2 - 24 \end{pmatrix} + u_1 \begin{pmatrix} 1 \\ 4 \end{pmatrix} + u_2 \begin{pmatrix} 2x_1 \\ -8 \end{pmatrix} + u_2 \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{62}$$

$$u_i \geq 0 \text{ for } i = 1, 2, 3 \tag{63}$$

$$u_1(x_1 + 4x_2 - 40) = 0 \tag{64}$$

$$u_2(x_1^2 - 8x_2) = 0 \tag{65}$$

$$u_3(-x_1) = 0 \tag{66}$$

The reader can verify that $\mathbf{x}^* = (8, 8)^t$ and $\mathbf{u}^* = (28/9, 5/9, 0)^t$ satisfy (59) through (66) and thus, as illustrated in Figure 11, the negative gradient of the objective at $\mathbf{x}^*$, $-\nabla f(\mathbf{x}^*)$, lies in the cone spanned by $\nabla g_1(\mathbf{x}^*)$ and $\nabla g_2(\mathbf{x}^*)$.

It should also be pointed out that if f and g_i for $i = 1, \ldots, m$ are convex functions and h_j for $j = 1, \ldots, p$ are linear functions, then the KKT conditions are also sufficient for optimality. This is the case in Example 8. In fact, these assumptions can be somewhat relaxed using the concepts of generalized convexity. For a detailed discussion of first- and second-order KKT conditions, see, for example, Fiacco and McCormick (1990, pp. 17–34).

The following section discusses a solution procedure that is a direct application of the KKT conditions.

4.3. Quadratic Programming

Quadratic programming problems are an important class of linearly constrained problems having the following form:

$$\text{(QP) Minimize } \mathbf{c}^t\mathbf{x} + 1/2\mathbf{x}^t\mathbf{H}\mathbf{x} \tag{67}$$

$$\text{Subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{68}$$

$$\mathbf{x} \geq \mathbf{0} \tag{69}$$

where $\mathbf{H}$ is an $n \times n$ symmetric matrix, $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{c} = (c_1, \ldots, c_n)^t$, $\mathbf{b} = (b_1, \ldots, b_m)^t$ and the decision vector is $\mathbf{x} = (x_1, \ldots, x_n)^t$. The following method is based on the Karush–Kuhn–Tucker conditions and reduces the quadratic programming problem to what is referred to as a linear complementarity problem.

Let $\mathbf{u} = (u_1, \ldots, u_m)^t$ and $\mathbf{v} = (v_1, \ldots, v_n)^t$ be the dual multipliers for constraints $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ and $-\mathbf{x} \leq \mathbf{0}$, respectively. Then, using matrix notation, the KKT conditions for (QP) can be expressed as follows:

$$\mathbf{A}\mathbf{x} + \mathbf{s} = \mathbf{b} \tag{70}$$

$$-\mathbf{A}^t\mathbf{u} - \mathbf{H}\mathbf{x} + \mathbf{v} = \mathbf{c} \tag{71}$$

$$\mathbf{x}, \mathbf{v} \geq \mathbf{0}, \mathbf{s}, \mathbf{u} \geq \mathbf{0} \tag{72}$$

$$\mathbf{x}^t\mathbf{v} = 0, \mathbf{u}^t\mathbf{s} = 0 \tag{73}$$

If $\mathbf{H}$ is positive semidefinite, then problem (QP) is a convex program. Thus, the KKT conditions are sufficient in this case, and any solution to this system will yield a global optimal solution to (QP). When $\mathbf{H}$ is indefinite, then local optimal solutions which are not global optimal solutions may occur.

The system (70)–(73) can be expressed in the form

$$\mathbf{w} - \mathbf{M}\mathbf{z} = \mathbf{q} \tag{74}$$

$$\mathbf{w} \geq \mathbf{0}, \mathbf{z} \geq \mathbf{0} \tag{75}$$

$$\mathbf{w}^t\mathbf{z} = 0 \tag{76}$$

where $\mathbf{M} = \begin{pmatrix} \mathbf{0} & -\mathbf{A} \\ \mathbf{A}^t & \mathbf{H} \end{pmatrix}$, $\mathbf{q} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix}$, $\mathbf{w} = \begin{pmatrix} \mathbf{s} \\ \mathbf{v} \end{pmatrix}$, $\mathbf{Z} = \begin{pmatrix} \mathbf{u} \\ \mathbf{x} \end{pmatrix}$.

Written in this form, (74)–(76) are an example of a *linear complementarity problem,* which also has applications in game theory. In this context, (76) is referred to as the *complementarity condition,* and all w_i, z_i pairs are said to be *complementary variables*. A method for finding a solution to this system is the complementary pivoting algorithm credited to Lemke (1968). Under certain assumptions on the matrix $\mathbf{M}$, the algorithm determines a solution or finds a direction indicating unboundedness in a finite number of iterations.

Other solution procedures for quadratic programming problems include conjugate gradient methods and the Dantzig–Wolfe method (see Dantzig 1963), which uses a modification of the simplex algorithm for linear programming.

4.4. Separable Programming

Separable programming is a procedure for obtaining an approximate solution to a nonlinear programming problem in which the objective function and constraint functions can be expressed as the sum of univariate functions. A separable programming problem has the following form:

$$\text{(SP) Minimize } f(\mathbf{x}) = \sum_{j=1}^{n} f_j(x_j) \tag{77}$$

$$\text{Subject to } \sum_{j=1}^{n} g_{ij}(x_j) \leq b_i \text{ for } i, \ldots, m \tag{78}$$

$$x_j \geq 0 \text{ for } j = 1, \ldots, n \tag{79}$$

Here, for example, $f(\mathbf{x})$ may represent the total cost whereas $f_j(x_j)$ represents the cost contribution of variable x_j.

The solution technique involves approximating each of the functions $f_j(x_j)$ and each of the functions $g_{ij}(x_j)$ by piecewise linear functions on their respective intervals of interest. Before presenting the resulting mathematical programming formulation, it is informative to review the idea of a linear approximation.

A *piecewise linear approximation* of a univariate function $h(x)$ on an interval of interest $[a, b]$ is illustrated in Figure 12. The interval $[a, b]$ is partitioned using t grid points, $a = x_1, x_2, \ldots, x_t = b$. Then the linear approximation, $l_k(x)$, on subinterval $[x_k, x_{k+1}]$ can be written as follows using the concept of a convex combination.

$$l_k(x) = \lambda h(x_k) + (1 - \lambda)h(x_{k+1}) \tag{80}$$

where

$$x = \lambda x_k + (1 - \lambda)x_{k+1} \tag{81}$$

$$\lambda \in [0, 1] \tag{82}$$

Generalizing, the piecewise linear approximation of $h(x)$ on the interval $[a, b]$ is given by

$$l(x) = \sum_{k=1}^{t} \lambda_k h(x_k) \tag{83}$$

$$\sum_{k=1}^{t} \lambda_k = 1 \tag{84}$$

$$\lambda_k \geq 0 \quad \text{for } k = 1, \ldots, t \tag{85}$$

where, at most, two adjacent λ_ks are positive. This last restriction arises because if nonadjacent λ_ks are positive, the resulting approximation will not lie on the piecewise linear approximating function. The accuracy of this approximation improves as the number of grid points increases, but, increasing the number of grid points increases the number of variables in the approximating problem.

Denoting the grid points for variable x_j by x_{kj} for $k = 1, \ldots, t_j$, the approximating problem is

$$\text{Minimize} \sum_{j=1}^{n} \sum_{k=1}^{t_j} \lambda_{kj} f_j(x_{kj}) \tag{86}$$

$$\text{Subject to} \sum_{j=1}^{n} \sum_{k=1}^{t_j} \lambda_{kj} g_{ij}(x_{kj}) = b_i \quad \text{for } i = 1, \ldots, m \tag{87}$$

$$\sum_{k=1}^{t_j} \lambda_{kj} = 1 \quad \text{for } j = 1, \ldots, n \tag{88}$$

$$\lambda_{kj} \geq 0 \quad \text{for } k = 1, \ldots, t_j; \quad j = 1, \ldots, n \tag{89}$$

$$\text{At most, two adjacent } \lambda_{kj}\text{s are positive for } j = 1, \ldots, n \tag{90}$$

Except for constraint (90), this is a linear programming problem in the variables λ_{kj}, which can be solved by the simplex method provided a restricted basis entry rule is used which enforces (90). However, in the case when each f_j is strictly convex and each g_{ij} is convex, constraint (90) can be

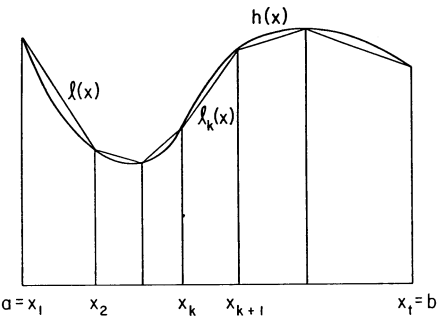


Figure 12 Piecewise Linear Approximation, $l(x)$, of a Function, $h(x)$.

neglected since it will be satisfied automatically by a solution generated by the simplex method. For additional details, see Miller (1963), and Wolfe (1963).

4.5. Geometric Programming

As has been shown in the two previous sections, it is sometimes possible to find the solution of a problem by transforming it into a simpler problem. This is also the case with geometric programming problems, another important class of nonlinear programming problems. The motivation for addressing this class of problems originated with work in engineering design. Geometric programming problems have the following general form:

$$\text{(GP) Minimize } g_0(\mathbf{x}) = \sum_{k=1}^{t_0} c_{0k} \prod_{j=1}^{n} (x_j)^{a_{0jk}} \tag{91}$$

$$\text{Subject to } g_i(\mathbf{x}) = \sum_{k=1}^{t_i} c_{ik} \prod_{j=1}^{n} (x_j)^{a_{ijk}} \leq 1 \text{ for } i = 1, \ldots, m \tag{92}$$

$$x_j > 0 \text{ for } j = 1, \ldots, n \tag{93}$$

where the coefficients c_{ik}, $i = 0, \ldots, m$ must be positive. Although each of the functions $g_i(\mathbf{x})$ for $i = 0, \ldots, m$ is not a true polynomial since the exponents a_{ijk} are not restricted to be integers, they are generally referred to as *posynomials* or positive generalized polynomials.

Problem (GP) has a nonlinear objective function and nonlinear constraints and, as such, is quite difficult to solve. However, geometric programming problems belong to a class of problems whose dual problems involve only linear constraints. Let δ_{ik} be the dual variable associated with the term $c_{ik}\prod_{j=1}^{n} (x_j)^{a_{ijk}}$. Then the dual problem for problem (GP) can be written as follows:

$$\text{Minimize } v(\boldsymbol{\delta}, \boldsymbol{\lambda}) = \prod_{i=0}^{m} \prod_{k=1}^{t_i} \left(\frac{c_{ik}}{\delta_{ik}}\right)^{\delta_{ik}} \prod_{i=1}^{m} (\lambda_i)^{\lambda_i} \tag{94}$$

$$\text{Subject to } \sum_{i=0}^{m} \sum_{k=1}^{t_i} a_{ijk}\delta_{ik} = 0 \text{ for } j = 1, \ldots, n \tag{95}$$

$$\sum_{k=1}^{t_0} \delta_{0k} = 1 \tag{96}$$

$$\sum_{k=1}^{t_i} \delta_{0k} = \lambda_i \text{ for } i = 1, \ldots, m \tag{97}$$

$$\boldsymbol{\delta} \geq \mathbf{0} \tag{98}$$

This dual problem has $t = \sum_{i=0}^{m} t_i$ variables, and at best, in the case when $t - n - 1 = 0$, determining the solution simply involves solving a square system of linear equations in nonnegative variables. The quantity $t - n - 1$ is referred to as the *degree of difficulty*. Although the dual objective function as written in (94) is neither convex nor concave, by using a natural logarithm transformation, the resulting function can be shown to be a concave function. In fact, by taking the logarithm, the objective function becomes separable and the dual problem can be cast as a separable programming problem. Thus, at worst, solving the dual problem involves maximizing a nonlinear concave objective function subject to linear constraints. As such, any local solution to the dual problem is a global solution. Even so, the advantage of solving the dual problem is offset by the fact that it can be difficult to find the optimal primal variables given the optimal dual variables and the optimal objective value.

Example 9

$$\text{Minimize } g_0(\mathbf{x}) = 4x_1x_2^3 + 20x_1^{-2} \tag{99}$$

$$\text{Subject to } g_1(\mathbf{x}) = x_1x_2^{-1} \leq 1 \tag{100}$$

$$x_1, x_2 > 0 \tag{101}$$

Letting δ_{01}, δ_{02}, and δ_{11} be the dual variables for the terms $4x_1x_2^3$, $20x_1^{-2}$, and $x_1x_2^{-1}$, respectively, then the dual problem becomes

$$\text{Maximize } v(\boldsymbol{\delta}, \boldsymbol{\lambda}) = \left(\frac{4}{\delta_{01}}\right)^{\delta_{01}} \left(\frac{20}{\delta_{02}}\right)^{\delta_{02}} \left(\frac{1}{\delta_{11}}\right)^{\delta_{11}} (\lambda_1)^{\lambda_1} \tag{102}$$

$$\text{Subject to } \delta_{01} - 2\delta_{02} + \delta_{11} = 0 \tag{103}$$

$$3\delta_{01} - \delta_{11} = 0 \tag{104}$$

$$\delta_{01} + \delta_{02} = 1 \tag{105}$$

$$\delta_{11} = \lambda_1 \tag{106}$$

$$\delta_{01}, \delta_{02}, \delta_{11} > 0 \tag{107}$$

Condition (103) is derived from the exponents of x_1 in the respective terms of the primal. Similarly, (104) is derived from the exponents of x_2. These are referred to as the *orthogonality conditions,* whereas (105) is termed the *normality condition*. In this case $t - n - 1 = 3 - 2 - 1 = 0$ and the dual problem can be solved by finding the solution of the linear system of equations, (103)–(105). Thus, $\delta_{01}^* = 1/3$, $\delta_{02}^* = 2/3$, $\lambda_1^* = \delta_{01}^* = 1$, with $g_0(\mathbf{x}^*) = v(\boldsymbol{\delta}^*, \boldsymbol{\lambda}^*) = 12^{1/3}30^{2/3}$.

The optimal values of the primal variables can be recovered by utilizing the following condition:

$$\delta_{ok}^* = \frac{c_{0k}}{g_0(\mathbf{x}^*)} \prod_{j=1}^{n} (x_j^*)^{a_{0jk}} \text{ for } k = 1, \ldots, t_0 \tag{108}$$

Substituting into (108) yields

$$\frac{1}{3} = \frac{4}{12^{1/3}30^{2/3}} x_1 x_2^3 \tag{109}$$

$$\frac{2}{3} = \frac{20}{12^{1/3}30^{2/3}} x_1^{-2} \tag{110}$$

Now solving (109) and (110) gives $x_1^* = x_2^* = (5/2)^{1/6}$.

For a complete discussion of geometric programming, the interested reader is referred to Duffin et al. (1967), and Beighter and Phillips (1976). Reviews of software for solving geometric programming problems are provided in Dembo (1976) and Rijckaert and Walraven (1985).

4.6. Methods of Feasible Directions

Consider the following nonlinear programming problem:

$$\text{Minimize } f(\mathbf{x}) \tag{111}$$

$$\text{Subject to } \mathbf{x} \in S \subseteq E_n \tag{112}$$

Conceptually, methods of feasible directions operate in a manner similar to unconstrained multidimensional search techniques. That is, the basic idea is, given a feasible point $\mathbf{x}_k$, determine a direction $\mathbf{d}_k$ and a step length α_k that yield the new point $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$. In the constrained case, however, care must to taken to choose a direction $\mathbf{d}_k$ that not only produces a point $\mathbf{x}_{k+1}$ that improves the objective function, but also maintains the feasibility of $\mathbf{x}_{k+1}$. For a differentiable objective function $f(\mathbf{x})$, an *improving feasible direction* $\mathbf{d}_k$ at the point $\mathbf{x}_k \in S$, has the following two properties (see Figure 13):

1. $\nabla f(\mathbf{x}_k)^t \mathbf{d}_k < 0$, that is, the *directional derivative* at $\mathbf{x}_k$ in the direction $\mathbf{d}_k$ is negative, resulting in a reduction in objective value. Geometrically, this means that $\mathbf{d}_k$ forms an acute angle with $-\nabla f(\mathbf{x}_k)$.
2. $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{d}_k \in S$ for some $\alpha > 0$, that is, a feasible move is possible in the direction $\mathbf{d}_k$.

Thus, methods of feasible directions operate by determining an improving feasible direction and then solving a line search problem to determine the step length in that direction. This process is repeated until some stopping rule is satisfied. Since the sequence of points generated is feasible to the primal problem, these are often called *primal methods*. The way in which the directions are generated and the step sizes are computed determines the various methods.

One such method that can be applied to a nonlinear programming problem with a linear constraint set is the method of reduced gradient, originally proposed by Wolfe (1963). It operates in a manner

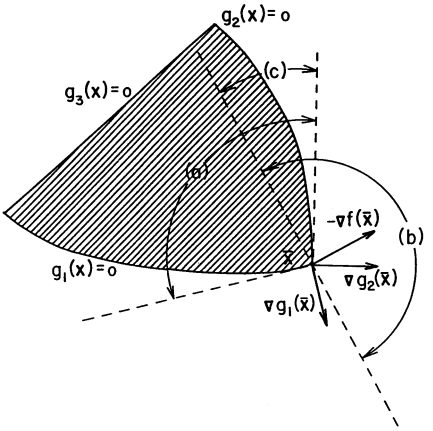


Figure 13 Illustration of (*a*) Feasible direction set at $\bar{\mathbf{x}}$, (*b*) Improving direction set at $\bar{\mathbf{x}}$, and (*c*) Improving feasible direction set at $\bar{\mathbf{x}}$.

similar to the simplex method for linear programming by using a set of independent variables to reduce the dimensionality of the problem. The reduced gradient is the gradient with respect to these independent variables. This method was generalized to handle nonlinear constraints by Abadie and Carpentier (1969) and is called the generalized reduced gradient method (GRG). There are several other methods of feasible directions, including those proposed by Zoutendijk (1960), the gradient projection method of Rosen (1961), and the convex simplex method of Zangwill (1967).

4.7. Sequential Unconstrained Minimization Techniques

In this section, methods for converting a general constrained nonlinear programming problem into an equivalent unconstrained problem are discussed. Once this conversion has been made, algorithms for unconstrained optimization can be applied. Unfortunately, there are computational difficulties associated with this process, and instead of solving a single unconstrained problem, it is usually necessary to solve a sequence of unconstrained problems. Although penalty functions methods were originally introduced by Courant (1943), the sequential unconstrained minimization technique (SUMT) was primarily developed by Fiacco and McCormick (1964). There are two basic approaches, both of which add a penalty term to the objective function. In the *penalty function method* (or exterior penalty function method), the optimal solution is approached by a sequence of infeasible points. That is, the optimum is approached from the exterior of the feasible region. In the second approach, known as the *barrier function method* (or interior penalty function method), the sequence of points generated converges to the optimum from within the feasible region.

4.7.1. Penalty Function Methods

Consider the following problem:

$$\text{Minimize } f(\mathbf{x}) \tag{113}$$

$$\text{Subject to } \mathbf{x} \in S \subseteq E_n \tag{114}$$

The basic idea is to approximate this problem with an unconstrained problem by adding a penalty function to the objective function that prescribes a high cost for violation of the constraint set S. This new unconstrained *auxiliary problem* is of the form

$$\text{Minimize } f(\mathbf{x}) + rP(\mathbf{x}) \tag{115}$$

where r is a positive constant, and $P(\mathbf{x})$ is chosen as a continuous penalty function such that (1) $P(\mathbf{x}) \geq 0$ for all $\mathbf{x}$, and (2) $P(\mathbf{x}) = 0$ if and only if $\mathbf{x} \in S$. For large r, the optimal solution of (115) will be in a region where $P(\mathbf{x})$ is small. However, if r is chosen too large, then the unconstrained problem becomes ill-conditioned and difficult to solve. Thus, a sequence of problems is solved in which r is increased from problem to problem. A sequence of problems that could be used for solving the general nonlinear programming problem (NLP), (46)–(48), is

$$\text{Minimize } f(\mathbf{x}) + r_k \left(\sum_{i=1}^{m} [\max\{0, g_i(\mathbf{x})\}]^q + \sum_{j=1}^{p} |h_j(\mathbf{x})|^q \right) \tag{116}$$

where q is a positive integer and r_k represents a strictly increasing sequence of positive numbers such that $r_k \to \infty$.

4.7.2. Barrier Function Methods

Barrier function methods are very similar to penalty function methods except that they start at an interior point of the feasible region and set a barrier against leaving the feasible region. In this case, the feasible region must have an interior, so this method is generally restricted to inequality constraints. Consider the nonlinear problem with inequality constraints,

$$\text{(NLPI) Minimize } f(\mathbf{x}) \tag{117}$$

$$\text{Subject to } g_i(\mathbf{x}) \leq 0 \text{ for } i = 1, \ldots, m \tag{118}$$

Ideally, a barrier function, $B(\mathbf{x})$, would assume the value zero for $\mathbf{x} \in \{\mathbf{x}: g_i(\mathbf{x}) < 0 \text{ for } i = 1, \ldots, m\}$ and the value ∞ on the boundary of the feasible region. $B(\mathbf{x})$ is usually defined such that (1) $B(\mathbf{x})$ is continuous and nonnegative on the interior of the feasible region, and (2) $B(\mathbf{x}) \to \infty$ on the boundary of the feasible region. A typical barrier function is

$$B(\mathbf{x}) = \sum_{i=1}^{m} [-1/g_i(\mathbf{x})] \tag{119}$$

This would result in the sequence of auxiliary problems

$$\text{Minimize } f(\mathbf{x}) + t_k \sum_{i=1}^{m} [-1/g_i(\mathbf{x})] \tag{120}$$

where t_k is a strictly decreasing sequence of positive numbers such that $t_k \to 0$.

4.7.3. Augmented Lagrangian Methods

In an attempt to avoid the ill-conditioning that occurs in the regular penalty and barrier function methods, Hestenes (1969) and Powell (1969) independently developed a multiplier method for solving nonlinearly constrained problems. This multiplier method was originally developed for equality constraints and involves optimizing a sequence of unconstrained augmented Lagrangian functions. It was later extended to handle inequality constraints by Rockafellar (1973).

Consider the mathematical programming problem

$$\text{(NLPE) Minimize } f(\mathbf{x}) \tag{121}$$

$$\text{Subject to } h_j(\mathbf{x}) = 0 \text{ for } j = 1, \ldots, p \tag{122}$$

The *augmented Lagrangian* function,

$$L_r(\mathbf{x}, \mathbf{v}) = f(\mathbf{x}) + \sum_{j=1}^{p} v_j h_j(\mathbf{x}) + r \sum_{j=1}^{p} (h_j(\mathbf{x}))^2 \tag{123}$$

is formed by introducing a multiplier vector, $\mathbf{v} = (v_1, \ldots, v_p)^t$ and a positive penalty parameter r. Note that $L_r(\mathbf{x}, \mathbf{v})$ in (123) is the Lagrangian function (37) augmented with the term $r \sum_{j=1}^{p} (h_j(\mathbf{x}))^2$. Thus, instead of a single penalty parameter, as in regular penalty function methods, the augmented Lagrangian requires estimates of the Lagrange multipliers. Using these multiplier estimates, which are updated from iteration to iteration, reduces the ill-conditioning of the unconstrained problems. Assuming that at iteration k, estimates $\mathbf{x}_k$, $\mathbf{v}_k$, and penalty parameter r_k are available, the problem minimize $L_{r_k}(\mathbf{x}, \mathbf{v}_k)$ is solved to find the local unconstrained solution $\mathbf{x}_{k+1}$. The updated vector $\mathbf{v}_{k+1}$ is then found using the relationship

$$\mathbf{v}_{k+1} = \mathbf{v}_k + 2r_k \mathbf{h}(\mathbf{x}_{k+1}) \tag{124}$$

where $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_p(\mathbf{x}))^t$. Upon selecting $r_{k+1} \geq r_k$, the process is repeated until $\mathbf{x}_{k+1}$ is sufficiently close to a local solution of (NLPE).

A summary of basic penalty function techniques, as well as exact penalty functions and multiplier methods are discussed in Fletcher (1987, pp. 277–304). For a complete discussion of multiplier methods, the interested reader is referred to Bertsekas (1982).

4.8. Successive Linear Programming

Successive (or sequential) linear programming (SLP) algorithms were introduced by Griffith and Stewart (1961) and have been used in a number of application areas, especially the oil and gas industry. SLP algorithms solve nonlinear optimization problems by using a sequence of linear programs, and computational results have shown they are particularly efficient on problems that are highly constrained. Consider the following nonlinear programming problem:

$$\text{(NLP1) Minimize } f(\mathbf{x}) + \mathbf{c}^t\mathbf{y} \tag{125}$$

$$\text{Subject to } \mathbf{g}(\mathbf{x}) + \mathbf{A}\mathbf{y} = \mathbf{b} \tag{126}$$

$$\mathbf{D}\mathbf{x} + \mathbf{E}\mathbf{y} = \mathbf{e} \tag{127}$$

$$\mathbf{l} \le \mathbf{x} \le \mathbf{u} \tag{128}$$

$$\mathbf{s} \le \mathbf{y} \le \mathbf{t} \tag{129}$$

where $\mathbf{A}$ is an $m \times p$ matrix, $\mathbf{D}$ is $q \times n$, $\mathbf{E}$ is $q \times p$, $\mathbf{c, s, t} \in E_p$, $\mathbf{l, u} \in E_n$, $\mathbf{b} \in E_m$, $\mathbf{e} \in E_q$, $f : E_n \to E_1$, and $\mathbf{g} : E_n \to E_m$, that is, $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_m(\mathbf{x}))^t$. Note that the decision variables have been partitioned into the nonlinear variables $\mathbf{x} \in E_n$, and the variables $\mathbf{y} \in E_p$, which appear only linearly. Similarly, the constraints are divided into nonlinear constraints (126) involving the vector of nonlinear differentiable functions $\mathbf{g}(\mathbf{x})$, and the linear constraints (127). The nonlinear variables $\mathbf{x}$ appear in the objective function (125) via the nonlinear differentiable function f.

The basic idea is, given a base point $\bar{\mathbf{x}}$, the nonlinear functions, f, $\mathbf{g}$, are linearized using first order Taylor series approximations. That is, $f(\mathbf{x})$ is replaced by $f(\bar{\mathbf{x}}) + \nabla f(\bar{\mathbf{x}})^t\mathbf{d}$ where $\mathbf{d} = \mathbf{x} - \bar{\mathbf{x}}$. Similarly, $g_i(\mathbf{x})$ for $i = 1, \ldots, m$, is replaced by $g_i(\bar{\mathbf{x}}) + \nabla g_i(\bar{\mathbf{x}})^t\mathbf{d}$. It is assumed that these linear approximations are accurate on some interval $-\boldsymbol{\delta} \le \mathbf{d} \le \boldsymbol{\delta}$ where $\boldsymbol{\delta} \in E_n$, $\boldsymbol{\delta} > \mathbf{0}$. Substituting these results into (NLP1) yields the linear program

$$\text{(LP}(\bar{\mathbf{x}}, \boldsymbol{\delta})\text{) Minimize } \nabla f(\bar{\mathbf{x}})^t\mathbf{d} + \mathbf{c}^t\mathbf{y} \tag{130}$$

$$\text{Subject to } \mathbf{J}(\bar{\mathbf{x}})\mathbf{d} + \mathbf{A}\mathbf{y} = \mathbf{b} - \mathbf{g}(\bar{\mathbf{x}}) \tag{131}$$

$$\mathbf{D}\mathbf{d} + \mathbf{E}\mathbf{y} = \mathbf{e} - \mathbf{D}\bar{\mathbf{x}} \tag{132}$$

$$\max(\mathbf{l} - \bar{\mathbf{x}}, -\boldsymbol{\delta}) \le \mathbf{d} \le \max(\mathbf{u} - \bar{\mathbf{x}}, \boldsymbol{\delta}) \tag{133}$$

$$\mathbf{s} \le \mathbf{y} \le \mathbf{t} \tag{134}$$

where $\mathbf{J}(\bar{\mathbf{x}})$ is the Jacobian matrix whose jth column is $\nabla g_j(\bar{\mathbf{x}})$.

Assuming a feasible solution exists, LP($\bar{\mathbf{x}}$, $\boldsymbol{\delta}$) is solved to find the solution $\bar{\mathbf{y}}$, $\bar{\mathbf{d}}$ that results in the candidate solution ($\bar{\mathbf{x}} + \bar{\mathbf{d}}$, $\bar{\mathbf{y}}$). If this solution is acceptable, then the bounds $\boldsymbol{\delta}$ may be increased and the process repeated. Otherwise, the bounds $\boldsymbol{\delta}$ are decreased and LP($\bar{\mathbf{x}}$, $\boldsymbol{\delta}$) is resolved. The process terminates when $\|\bar{\mathbf{d}}\|$ is sufficiently small.

Details of SLP algorithms along with computational results are contained in Palacios-Gomez et al. (1982), Baker and Lasdon (1985), and Zhang et al. (1985).

4.9. Successive Quadratic Programming

Successive quadratic programming (SQP) algorithms are an important class of methods that has shown much promise in solving general nonlinear programming problems. The methods are also referred to as Wilson–Han–Powell-type methods (Wilson 1963; Han 1976; Powell 1978) as well as Lagrange–Newton methods. SQP algorithms essentially determine a Karush–Kuhn–Tucker point by applying Newton's method to find a stationary point of the Lagrangian function. For a discussion of SQP algorithms, see, for example, Stoer (1985) and Fletcher (1987, pp. 304–317).

4.10. Nonsmooth Optimization

Nonsmooth or nondifferentiable optimization plays an important role in large-scale programming and addresses mathematical programming problems in which the functions involved have discontinuous first derivatives. Thus, classical methods that rely on gradient information fail to solve these problems, and alternative nonstandard approaches must be used. These alternative methods include subgradient methods and bundle methods. The interested reader is referred to Shor (1985), Zowe (1985), and Fletcher (1987, pp. 357–414).

5. ONLINE SOURCES OF INFORMATION ON OPTIMIZATION

There is a wealth of information on mathematical programming available on the World Wide Web. The resources range from electronic books on optimization to libraries of source code for optimization problems to test data archives for mathematical programming. The following is a listing of a few of the more comprehensive optimization sites. Many of the sites also provide links to other related sites.

- Center for Advanced Modeling and Optimization (CAMO)
  http://www.ici.ro/camo/
- Mathematical Optimization (Computational Science Education Project)
  http://csep1.phy.ornl.gov/mo/mo.html
- The Optimization Technology Center and The Network Enabled Optimization System (NEOS)
  http://www-fp.mcs.anl.gov/otc/
- Nonlinear Programming FAQ (Optimization Technology Center)
  http://www-unix.mcs.anl.gov/otc/Guide/faq/nonlinear-programming-faq.html
- Mathematical Programming Glossary (Harvey J. Greenberg)
  http://www.cudenver.edu/~hgreenbe/glossary/glossary.html

6. NONLINEAR PROGRAMMING CODES

Due to advances in computer technology and algorithmic techniques, mathematical programming codes have made significant progress in recent years. Below is a partial listing of some of the nonlinear programming software that is available to the industrial engineering practitioner. For a detailed discussion of software packages, the interested reader is referred to Moré and Wright (1993). Several online sites also provide listings and reviews of available optimization software. See, for example,

- Decision Tree for Optimization Software (H. D. Mittelmann and P. Spellucci)
  http://plato.la.asu.edu/topics/problems/nlores.html
- Guide to Available Mathematical Software (National Institute of Standards and Technology)
  http://gams.cam.nist.gov/
- Nonlinear Programming FAQ (Optimization Technology Center)
  http://www-unix.mcs.anl.gov/otc/Guide/faq/nonlinear-programming-faq.html
- Nonlinear Programming Packages (Center for Advanced Modeling and Optimization)
  http://www.ici.ro/camo/hnp.htm
- Optimization Software (Optimization Technology Center)
  http://www-fp.mcs.anl.gov/otc/Guide/SoftwareGuide/

6.1. Optimization Software

CONOPT
Problem Type: Nonlinear programs with sparse nonlinear constraints
Method: Generalized reduced gradient
Author: Arne S. Drud, ARKI Consulting and Development A/S, Denmark
Contact: ARKI Consulting and Development A/S, Email: info@arki.dk

DONLP2
Problem Type: Smooth nonlinear functions subject to smooth constraints
Method: Sequential quadratic programming
Author: Peter Spellucci, Technical University Darmstadt, Germany
Contact: http://www.mathematik.tu-darmstadt.de/ags/ag8/spellucci/

EA3
Problem Type: Nonlinear programs
Method: Ellipsoid algorithm
Authors: J. G. Ecker, M. Kupferschmid, Rensselaer Polytechnic Institute, NY
Contact: J. G. Ecker, Department of Mathematical Sciences; M. Kupferschmid, Alan M. Voorhees Computing Center, Rensselaer Polytechnic Institute, Troy, NY 12181

FSQP
Problem Type: Multiple linear/nonlinear objective functions with linear/nonlinear constraints
Method: Sequential quadratic programming
Authors: Eliane R. Panier, Andre Tits, Jian Zhou, Craig Lawrence, University of Maryland

Contact: http://www.isr.umd.edu/Labs/CACSE/FSQP/fsqp.html

GRG2
Problem Type: Nonlinear programs
Method: Generalized reduced gradient
Author: Prof. Leon Lasdon, The University of Texas at Austin
Contact: http://www.optimalmethods.com/

LANCELOT
Problem Type: Large-scale optimization problems
Method: Penalty method
Authors: Andy Conn, IBM T. J. Watson Research Center, NY, Nick Gould, Rutherford Appleton Laboratory, UK, Philippe Toint, Facultés Universitaires Notre Dame de la Paix, Belgium
Contact: http://www.cse.clrc.ac.uk/Activity/LANCELOT+165

LSGRG2
Problem Type: Large-scale nonlinear programs
Method: Generalized reduced gradient
Author: Prof. Leon Lasdon, The University of Texas at Austin
Contact: http://www.optimalmethods.com/

MINOPT
Problem Type: Linear, mixed-integer, nonlinear, dynamic, and mixed-integer nonlinear programs
Method: Generalized benders decomposition, outer approximation and variants, generalized cross decomposition
Authors: C. Schweiger, Christodoulos A. Floudas, Princeton University
Contact: http://titan.princeton.edu/MINOPT/minopt.html

MINOS
Problem Type: Large-scale linear and nonlinear programs
Method: Projected Lagrangian
Authors: Bruce A. Murtagh, University of New South Wales, Australia, Michael A. Saunders, Stanford University
Contact: http://www.stanford.edu/~saunders/brochure/brochure.html

NIMBUS
Problem Type: Differentiable/nondifferentiable multiobjective/single objective optimization problems with nonlinear/linear constraints
Method: Nondifferentiable interactive multiobjective bundle-based optimization
Authors: Kaisa Miettinen, University of Jyväskylä, Finland
Contact: http://nimbus.mit.jyu.fi/

NLPQL
Problem Type: Nonlinear programs
Method: Sequential quadratic programming
Authors: K. Schittkowski, University of Bayreuth, Germany
Contact: http://www.uni-bayreuth.de/departments/math/~kschittkowski/nlpql.htm

NLPQLB
Problem Type: Smooth nonlinear programming with many constraints
Method: Sequential quadratic programming
Authors: Schittkowski, University of Bayreuth, Germany
Contact: http://www.uni-bayreuth.de/departments/math/~kschittkowski/nlpqlb.htm

NPSOL
Problem Type: Dense linear and nonlinear programs
Method: Sequential quadratic programming
Authors: Philip Gill, University of California, San Diego; Walter Murray, Michael A. Saunders, Stanford University; Margaret H. Wright, AT&T Bell Laboratories
Contact: http://www.stanford.edu/~saunders/brochure/brochure.html

OPTIMA Library
Problem Type: Unconstrained and constrained nonlinear optimization
Method: Various methods
Authors: M. C. Bartholomew-Biggs, University of Hertfordshire, United Kingdom
Contact: Dr. M. C. Bartholomew-Biggs, Numerical Optimisation Center, Hatfield, Hertfordshire AL10 9AB, United Kingdom

SNOPT
Problem Type: Large-scale linear and nonlinear programs
Method: Sparse sequential quadratic programming
Authors: Philip Gill, University of California, San Diego; Walter Murray, Michael A. Saunders, Stanford University
Contact: http://www.stanford.edu/~saunders/brochure/brochure.html

SOLVOPT
Problem Type: Nonlinear programs
Method: Exact penalty method
Authors: Alexei V. Kuntsevich, Karl-Franzens Universität Graz, Austria, Franz Kappel
Contact: http://bedvgm.kfunigraz.ac.at:8001/alex/solvopt/

SPENBAR
Problem Type: Nonlinear programs
Method: Modified penalty method
Authors: Neculai Andrei, Research Institute for Informatics, Romania
Contact: Neculai Andrei, Research Institute for Informatics, 8-10, Bdl. Maresal Averescu, 71316 Bucharest, Romania, E-mail: nandrei@u3.ici.ro

TRON
Problem Type: Large bound-constrained optimization problems
Method: Trust region Newton method
Authors: Chih-Jen Lin,National Taiwan University; Jorge Moré, Argonne National Laboratory
Contact: http://www-unix.mcs.anl.gov/~more/tron/

REFERENCES

Abadie, J., and Carpentier, J. (1969), "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints," in *Optimization,* R. Fletcher, Ed., Academic Press, New York.

Armijo, L. (1966), "Minimization of Functions having Lipschitz Continuous First-Partial Derivatives," *Pacific J. Mathematics,* Vol. 16, No. 1, pp. 1–3.

Avriel, M. (1976), *Nonlinear Programming: Analysis and Methods,* Prentice-Hall, Englewood Cliffs, NJ, pp. 155–183.

Baker, T. E., and Lasdon, L. S. (1985), "Successive Linear Programming at Exxon," *Management Science,* Vol. 31, No. 3, pp. 264–274.

Bazaraa, M. S., Sherali, H. D, and Shetty, C. M. (1994), *Nonlinear Programming: Theory and Applications,* John Wiley & Sons, New York.

Beightler, C. S., and Phillips, D. T. (1976), *Applied Geometric Programming,* John Wiley & Sons, New York.

Bertsekas, D. P. (1982), *Constrained Optimization and Lagrange Multiplier Methods,* Academic Press, New York, 1982.

Bracken, J., and McCormick, G. P. (1968), *Selected Applications of Nonlinear Programming,* John Wiley & Sons, New York.

Broyden, C. G. (1970), "The Convergence of a Class of Double Rank Minimization Algorithms 2. The New Algorithm," *Journal of Institute of Mathematics and Its Applications,* Vol. 6, pp. 222–231.

Courant, R. (1943), "Variational Methods for the Solution of Problems of Equilibrium and Vibration," *Bulletin of the American Mathematical Society,* Vol. 49, pp. 1–23.

Dantzig, G. B. (1963), *Linear Programming and Extensions,* Princeton University Press, Princeton, NJ, pp. 490–496.

Davidon, W. C. (1959), "Variable Metric Method for Minimization," *AEC Research Development Report,* ANL-5990.

Dembo, R. S. (1976), "The Current State of the Art of Algorithms and Computer Software for Geometric Programming," Working Paper No. 88, School of Organization and Management, Yale University, New Haven, CT.

Duffin, R. J., Peterson, E. L., and Zener, C. (1963), *Geometric Programming: Theory and Application,* John Wiley & Sons, New York.

Fiacco, A. V., and McCormick, G. P. (1964), "The Sequential Unconstrained Minimization Technique for Nonlinear Programming, A Primal-Dual Method," *Management Science,* Vol. 10, pp. 360–366.

Fiacco, A. V., and McCormick, G. P. (1990), *Nonlinear Programming: Sequential Unconstrained Minimization Techniques,* SIAM, Philadelphia.

Fletcher, R. (1970), "A New Approach to Variable Metric Algorithms," *Computer Journal,* Vol. 13, pp. 317–322.

Fletcher, R. (1987), *Practical Methods of Optimization,* 2nd Ed., John Wiley & Sons, New York.

Fletcher, R., and Powell, M. (1963), "A Rapidly Convergent Descent Method for Minimization," *Computer Journal,* Vol. 6, pp. 163–168.

Fletcher, R., and Reeves, C. M. (1964), "Function Minimization by Conjugate Gradients," *Computer Journal,* Vol. 7, pp. 149–154.

Goldfarb, D. (1970), "A Family of Variable Metric Methods Derived by Variational Means," *Mathematics of Computation,* Vol. 24, pp. 23–26.

Griffith, R. E., and Stewart, R. A. (1961), "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems," *Management Science,* Vol. 7, pp. 379–392.

Han, S. P. (1976), "Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems," *Mathematical Programming,* Vol. 11, pp. 263–282.

Hestenes, M. R. (1969), "Multiplier and Gradient Methods," *Journal of Optimization Theory and Applications,* Vol. 4, pp. 303–320.

Hooke, R., and Jeeves, T. A. (1961), "Direct Search Solution of Numerical and Statistical Problems," *J. Association Computer Machinery*, Vol. 8, pp. 212–229.

Karmarkar, N. (1984), "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorics,* Vol. 4, pp. 373–395.

Karush, W. (1939), "Minima of Functions of Several Variables with Inequalities as Side Conditions," M.S. Thesis, Department of Mathematics, University of Chicago.

Kiefer, J. (1957), "Optimal Sequential Search and Approximation Methods under Minimum Regularity Conditions," *SIAM Journal of Applied Mathematics,* Vol. 5, pp. 105–136.

Kuhn, H. W., and Tucker, A. W. (1951), "Nonlinear Programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability,* J. Neyman, Ed., University of California Press, Berkeley, pp. 481–492.

Lemke, C. E. (1968), "On Complementary Pivot Theory," in *Mathematics of the Decision Sciences,* G. B. Dantzig and A. F. Veinott, Eds., American Mathematical Society, Providence, RI.

Levenberg, K. (1944), "A Method for the Solution of Certain Nonlinear Problems in Least Squares," *Quart. Appl. Math*., Vol. 2, pp. 164–168.

Marqardt, D. W. (1963), "An Algorithm for the Least Squares Estimation of Nonlinear Parameters," *SIAM Journal,* Vol. 11, pp. 431–441.

McCormick, G. P. (1983), *Nonlinear Programming: Theory, Algorithms, and Applications,* John Wiley & Sons, New York, pp. 143–167.

Miller, C. E. (1963), "The Simplex Method for Local Separable Programming," in *Recent Advances in Mathematical Programming,* R. L. Graves and P. Wolfe, Eds., McGraw-Hill, New York.

Moré, J. J., and Wright, S. J. (1993), *Optimization Software Guide,* SIAM, Philadelphia.

Nelder, J. A., and Mead, R. (1964), "A Simplex Method for Function Minimization," *Computer Journal,* Vol. 7, pp. 308–313.

Palacios-Gomez, F., Lasdon, L., and Engquist, M. (1982), "Nonlinear Optimization by Successive Linear Programming," *Management Science,* Vol. 28, No. 10, pp. 1106–1120.

Powell, M. J. D. (1969), "A Method for Nonlinear Constraints in Minimization Problems," in *Optimization,* R. Fletcher, Ed., Academic Press, London.

Powell, M. J. D. (1978), "A Fast Algorithm for Nonlinearly Constrained Optimization Calculation," in *Numerical Analysis,* Lecture Notes in Mathematics, Vol. 630, G. A. Watson, Ed., Springer, Berlin.

Rijckaert, M. J., and Walraven, E. J. C. (1985), "Reflections on Geometric Programming," in *Computational Mathematical Programming,* K. Schittkowski, Ed., Springer, Berlin.

Rockafellar, R. T. (1973), "The Multiplier Method of Hestenes and Powell Applied to Convex Programming," *Journal of Optimization Theory and Applications,* Vol. 12, pp. 555–562.

Rosen, J. B. (1961), "The Gradient Projection Method for Nonlinear Programming Part II: Nonlinear Constraints," *SIAM Journal of Applied Mathematics,* Vol. 9, pp. 514–553.

Rosenbrock, H. H. (1960), "An Automatic Method for Finding the Greatest or Least Value of a Function," *Computer Journal,* Vol. 3, pp. 175–184.

Shanno, D. F. (1970), "Conditioning of Quasi-Newton Methods for Function Minimization," *Mathematics of Computation,* Vol. 24, pp. 647–656.

Shor, N. Z. (1985), *Minimization Methods for Non-differentiable Functions,* Springer, Berlin.

Spendley, W., Hext, G. R., and Himsworth, F. R. (1962), "Sequential Application of Simplex Designs of Optimization and Evolutionary Operations," *Techometrics,* Vol. 4, pp. 441–461.

Stoer, J. (1985), "Principals of Sequential Quadratic Programming Methods for Solving Nonlinear Programs," in *Computational Mathematical Programming,* K. Schittkowski, Ed., Springer, Berlin.

Wilson, R. B. (1963), "A Simplicial Algorithm for Concave Programming," Ph.D. Thesis, Graduate School of Business Administration, Harvard University, Cambridge, MA.

Wolfe, P. (1963), "Methods of Nonlinear Programming," in *Recent Advances in Mathematical Programming,* R. L. Graves and P. Wolfe, Eds., McGraw-Hill, New York.

Zangwill, W. I. (1967), "The Convex Simplex Method," *Management Science,* Vol. 14, pp. 221–283.

Zhang, J., Kim, N., and Lasdon, L. (1985), "An Improved Successive Linear Programming Algorithm," *Management Science,* Vol. 31, No. 10, pp. 1312–1331.

Zoutendijk, G. (1960), *Methods of Feasible Directions,* Elsevier, Amsterdam.

Zowe, J. (1985), "Nondifferentiable Optimization," in *Computational Mathematical Programming,* K. Schittkowski, Ed., Springer, Berlin.

ADDITIONAL READING

Avriel, M., *Nonlinear Programming: Analysis and Methods,* Prentice-Hall, Englewood Cliffs, NJ, 1976.

Bazarra, M. S., Sherali, H. D., and Shetty, C. M., *Nonlinear Programming: Theory & Applications,* John Wiley & Sons, New York, 1994.

Beightler, C. S., Phillips, D. T., and Wilde, D. J., *Foundations of Optimization,* 2nd Ed., Prentice-Hall, Englewood Cliffs, NJ, 1976.

Bertsekas, D. P., *Nonlinear Programming,* 2nd Ed., Athena Scientific, Belmont, MA.

Coleman, T. F., and Li, Y., *Large Scale Numerical Optimization,* SIAM, Philadelphia, 1990.

Dantzig, G. B., *Linear Programming and Extensions,* Princeton University Press, Princeton, NJ, 1963.

Du, D., and Sun, J., *Advances in Optimization and Approximation,* Kluwer, Dordrecht, 1994.

Duffin, R. J., Peterson, E. L., and Zener, C., *Geometric Programming,* John Wiley & Sons, New York.

Ecker, J. G., and Kupferschmid, M., *Introduction to Operations Research,* John Wiley & Sons, New York, 1988.

Fiacco, A. V., and McCormick, G. P., *Nonlinear Programming: Sequential Unconstrained Minimization Techniques,* SIAM, Philadelphia, 1990.

Fletcher, R., *Practical Methods of Optimization,* 2nd Ed., John Wiley & Sons, New York, 1987.

Floudas, C. A., *Deterministic Global Optimization: Theory, Algorithms and Applications,* Kluwer, Dordrecht, 1999.

Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization,* Academic Press, London, 1981.

Hillier, F. S., and Lieberman, G. J., *Introduction to Operations Research,* 6th Ed., McGraw-Hill, New York, 1995.

Lasdon, L. S., *Optimization Theory for Large Systems,* Macmillan, New York, 1970.

Luenberger, D. G., *Introduction to Linear and Nonlinear Programming,* 2nd Ed., Addison-Wesley, Reading, MA, 1984.

Mangasarian, O. L., *Nonlinear Programming,* McGraw-Hill, New York, 1969.

McCormick, G. P., *Nonlinear Programming: Theory, Algorithms, and Applications,* John Wiley & Sons, New York, 1983.

Minoux, M., *Mathematical Programming: Theory and Algorithms,* John Wiley & Sons, New York, 1986.

Nash, S. G., and Sofer, A., *Linear and Nonlinear Programming,* McGraw-Hill, New York, 1996.

Nocedal, J., and Wright, S. J., *Numerical Optimization,* Springer, New York, 1999.

Shapiro, J. F., *Mathematical Programming: Structures and Algorithms,* John Wiley & Sons, New York, 1979.

Zangwill, W. I., *Nonlinear Programming: A Unified Approach,* Prentice-Hall, Englewood Cliffs, NJ, 1969.