Chapter 10

SURVEY OF DESCENT BASED METHODS FOR UNCONSTRAINED AND LINEARLY CONSTRAINED MINIMIZATION

Nonlinear Programming Problems

Eventhough the title "Nonlinear Programming" may convey the impression that the subject includes all optimization problems other than linear programming problems, it is not usually the case. Optimization problems involving discrete valued variables (i. e., those which are restricted to assume values from specified discrete sets, such as 0-1 variables) are not usually considered under nonlinear programming, they are called **discrete**, or **mixed-discrete optimization problems** and studied separately. There are good reasons for this. To solve discrete optimization problems we normally need very special techniques (typically of some enumerative type) different from those needed to tackle continous variable optimization problems. So, the term nonlinear program usually refers to an optimization problem in which the variables are continuous variables, and the problem is of the following general form:

$$\begin{array}{lrll} \text{minimize} & \theta(x) & & \\ \text{subject to} & h_i(x) & = 0, & i = 1 \text{ to } m \\ & g_p(x) & \geqq 0, & p = 1 \text{ to } t \end{array} \tag{P}$$

where $\theta(x)$, $h_i(x)$, $g_p(x)$ are all real valued continuous functions of $x = (x_1, \ldots, x_n) \in \mathbf{R}^n$.

Suppose some of these functions are not differentiable at some points x. Assume that gradients exist for each function almost everywhere, but are not continuous. Then problem (P) is known as a **non-smooth** or **non-differentiable optimization problem**. On such a problem, the usual gradient-based methods and results may fail, and special attention must be given to the surfaces of non-differentiability, it becomes very important to consider generalized gradients to handle such problems.

If all the functions $\theta(x)$, $h_i(x)$, $g_p(x)$ are continuously differentiable, problem (P) is known as a **smooth nonlinear program**. In this book we only study smooth nonlinear programs. However, some of the techniques that we discuss may convert a smooth NLP into a special type of nonsmooth NLP, and then solve it. As an example, the simplicial method discussed in Section 2.7.6 to solve the smooth NLP: minimize $\theta(x)$, subject to $g_i(x) \leqq 0$, converts it into the NLP: minimize $\theta(x)$, subject to $s(x) \leqq 0$, where $s(x) = \max \cdot \{g_1(x), g_2(x), \dots, g_m(x)\}$. This modified problem is a nonsmooth optimization problem, since $s(x)$ may not be differentiable at some points x. However, because of the special nature of $s(x)$, we know that $\partial s(x) =$ convex hull of $\{\nabla g_i(x) : i$ such that $g_i(x) = s(x)\}$, and hence for any given x, it is easy to find at least one point in $\partial s(x)$, and the special simplicial algorithms discussed in Section 2.7, are able to solve this modified problem using only this information.

Consider the NLP (P) and assume that all the functions are continuously differentiable. The constraints in (P) are either equality constraints, or inequality constraints. (P) is the general form of the problem, and in a particular instance of (P), there may or may not be such constraints. This problem is said to be:

an **unconstrained minimization problem**, if there are no constraints on the variables, in the statement of the problem,

a **linear programming problem**, if all the functions $\theta(x)$, $h_i(x)$, $g_p(x)$ are affine functions,

a **quadratic programming problem**, if $\theta(x)$ is a quadratic function, and all $h_i(x)$ and $g_p(x)$ are affine functions,

an **equality constrained problem**, if there are no inequality constraints on the variables,

a **linearly constrained** NLP, if all the constraint functions $h_i(x)$, $g_p(x)$ are affine functions,

a **convex programming problem** if $\theta(x)$ is a convex function, all $h_i(x)$ are affine functions, and all $g_p(x)$ are concave functions,

a **nonconvex programming problem**, if it is not a convex programming problem as defined above.

In this chapter, we provide a brief survey of some commonly used algorithms for smooth NLPs, those in the areas of unconstrained and linearly constrained NLPs, which constitute alternate methods to those discussed so far for solving quadratic programs.

10.1 A FORMULATION EXAMPLE FOR A LINEARLY CONSTRAINED NONLINEAR PROGRAM

We begin this chapter with a practical example due to C. H. White, of a nonlinear model in which the constraints are linear. It arose in the boiler shop of a company which has five (5) boilers operating in parallel for generating steam. Data on the boilers is given below.

Tableau 10.1

Boiler i	Boiler load range limits					
	lower l_i	upper k_i	a_{0i}	a_{1i}	a_{2i}	a_{3i}
1	10 units	60	56.49	1.67	−.041	.00030
2	10	60	71.37	0.61	−.016	.00011
3	15	120	23.88	2.05	−.024	.00009
4	12.5	112.5	17.14	2.73	−.035	.00014
5	15	135	72.38	0.34	−.003	.00001

The unit measures the rate at which steam is produced per unit time. If the ith boiler is kept on, it must be operated within its load range limits l_i, k_i. The boiler's energy efficiency defined as a percentage is 100 × (energy content of output steam)/(energy content in the input fuel). It tends to increase as the load moves up from the minimum allowable operating load, and then peaks and drops as the load approaches the upper limit. Data was collected on the boiler efficiencies at different operating load levels, and the plots indicated that boiler efficiency can be approximated very well by a cubic polynomial of the operating load. Let $y(\xi)$ = efficiency of a boiler when it is operating at load ξ units. We approximate $y(\xi)$ by $f(\xi) = a_0 + a_1\xi + a_2\xi^2 + a_3\xi^3$, where a_0, a_1, a_2, a_3 are parameters to be estimated from data. The problem of determining the best values of the parameters that give the closest fit between observed efficiency and the cubic polynomial, is known as the **parameter estimation problem** or the **curve fitting problem**. Suppose we have r observations on a boiler, at load levels ξ_t, $t = 1$ to r yielding observed efficiencies of y_t, $t = 1$ to r respectively. To derive the closest fit we need to construct a measure of deviation of the functional value $f(\xi)$ from the observed $y(\xi)$ over the range of values of ξ used in the experiment, depending on the parameter vector $a = (a_0, a_1, a_2, a_3)$. Three different measures are in common use.

They are

$$L_2(a) = \sum_{t=1}^{r}(y_t - a_0 - \sum_{s=1}^{3} a_s \xi_t^{\mathbf{s}})^{\mathbf{2}}$$

$$L_1(a) = \sum_{t=1}^{r} |y_t - a_0 - \sum_{s=1}^{3} a_s \xi_t^{\mathbf{s}}|$$

$$L_\infty(a) = \text{Maximum } \{|y_t - a_0 - \sum_{s=1}^{3} a_s \xi_t^{\mathbf{s}}| : t = 1 \text{ to } r\}.$$

Since the $L_2(a)$ measure is a sum of squares, the technique which chooses the parameter vector a to minimize $L_2(a)$ is called the **least squares approach** or the **method of least squares**. If $\hat{a} = (\hat{a}_0, \hat{a}_1, \hat{a}_2, \hat{a}_3)$ is the best vector of parameter values obtained under this method, the function $\hat{a}_0 + \hat{a}_1\xi + \hat{a}_2\xi^{\mathbf{2}} + \hat{a}_3\xi^{\mathbf{3}}$ is called the **least squares approximation** for $y(\xi)$.

If the parameter vector a is determined so as to minimize the measure $L_\infty(a)$, the resulting function $f(\xi)$ is known as the **Tschebycheff approximation** for $y(\xi)$.

If all the parameters appear linearly in the functional form $f(\xi)$ (as in this boiler efficiency example) the problem of minimizing either the L_1- or L_∞-measures can both be posed as linear programs and solved by the efficient simplex method. However, if the parameters appear nonlinearly in the functional form, the least squares method is preferred for parameter estimation.

If the measure of deviation is too large even at the best parameter values, it is necessary to review the choice of the functional form and modify it. Besides, it is possible that no simple function provides a good approximation for all possible values of load. It is only necessary to find a good functional representation of the efficiency in the neighborhood of the optimum load values, if some reliable practical knowledge is available on the likely location of this optimum.

Thus, even the process of constructing a mathematical model for the problem might itself need the application of optimization algorithms for parameter estimation.

The Basic Difference Between Linear and Nonlinear Models

To construct a linear programming model involving n nonnegative variables subject to m constraints, we need to estimate the $(m+1)(n+1) - 1$ coefficients of the variables in the constraints and the objective function, these are the data elements in the model. Real life LP applications routinely involve models with $n = 100,000$ or more, and m as large as 6000. A large scale LP model is usually of this size.

To construct a nonlinear model, we have to determine the functional form of the objective and each constraint function, and obtain the best values for any parameters in each. For this reason, practical nonlinear models tend to have fewer variables than linear models. Depending on how complicated the functions involved are, a nonlinear model with about 200 variables could usually be considered as a large scale model.

Boiler Example, Continued

For the boiler problem, estimates of the best parameter values in the functional form for the efficiency of each boiler are given in Tableau 10.1.

At a point of time, the Company's steam requirements are 350 units per unit time. The problem is to determine how this total load of 350 units should be shared across the five (5) parallel boilers so as to minimize the total fuel cost. It may be possible to get a lower overall cost by shutting down one or more of the boilers and meeting the demand using only the remaining boilers. For example, here it can be verified that the total load of 350 units can be met using boilers 3, 4, and 5 only. Thus the problem of determining the most efficient plan to meet a load of exactly 350 units, leads to a mixed integer nonlinear programming problem in which there are five zero-one variables to determine which of the five boilers are shut down and which are kept operating, and the operating load level for the boilers that are kept operating. In this plant however, it is known that the Company's steam requirements vary with time. When the demand for steam goes up, if a boiler is kept operating, it is a relatively easy matter to increase the boiler's steam output by turning a few valves. On the other hand turning on a shut down boiler is an expensive operation. In order to be able to meet the varying steam requirements over time, it was determined that all the five boilers should be kept operating. Under this condition, since $x_i/f_i(x_i)$ is a measure of the energy cost of obtaining a load of x_i units from boiler i, we are lead to the following nonlinear model:

$$\begin{aligned} &\text{minimize } \sum_{i=1}^{5} x_i/f_i(x_i) \\ &\text{subject to } \sum_{i=1}^{5} x_i = 350 \\ &l_i \leqq x_i \leqq k_i, \; i = 1 \text{ to } 5 \end{aligned}$$

which is a linearly constrained nonlinear program.

Exercise

10.1 Using the $0 - 1$ variables y_i defined by

$$\begin{aligned} y_i &= 1 \quad \text{if the } i\text{th boiler is kept operating} \\ &= 0 \quad \text{otherwise} \end{aligned}$$

formulate the problem of determining the most efficient plan for producing exactly 350 units of steam per unit time as a mixed integer NLP.

10.2 TYPES OF SOLUTIONS FOR A NONLINEAR PROGRAM

Consider a NLP in which a function $\theta(x)$ is required to be optimized subject to some constraints on the variables $x = (x_1, \ldots, x_n)^T$. Let $\mathbf{K}$ denote the set of feasible solutions for this problem. For this problem a feasible solution $\overline{x} \in \mathbf{K}$ is said to be a

local minimum, if there exists an $\varepsilon > 0$ such that $\theta(x) \geqq \theta(\overline{x})$ for all $x \in \mathbf{K} \cap \{x : \|x - \overline{x}\| < \varepsilon\}$,

strong local minimum, if there exists an $\varepsilon > 0$ such that $\theta(x) > \theta(\overline{x})$ for all $x \in \mathbf{K} \cap \{x : \|x - \overline{x}\| < \varepsilon\}$, $x \neq \overline{x}$,

weak local minimum, if it is a local minimum, but not a strong one,

global minimum, if $\theta(x) \geqq \theta(\overline{x})$ for all $x \in \mathbf{K}$,

local maximum, if there exists an $\varepsilon > 0$ such that $\theta(x) \leqq \theta(\overline{x})$ for all $x \in \mathbf{K} \cap \{x : \|x - \overline{x}\| < \varepsilon\}$,

strong local maximum, if there exists an $\varepsilon > 0$ such that $\theta(x) < \theta(\overline{x})$ for all $x \in \mathbf{K} \cap \{x : \|x - \overline{x}\| < \varepsilon\}$, $x \neq \overline{x}$,

weak local maximum, if it is a local maximum, but not a strong one,

global maximum, if $\theta(x) \leqq \theta(\overline{x})$ for all $x \in \mathbf{K}$,

stationary point, if some necessary optimality conditions for the problem are satisfied at the point $\overline{x}$.

These concepts are illustrated in Figure 10.1 for the one dimensional problem: optimize $\theta(x)$ subject to $x \in \mathbf{R}^1$, $a \leqq x \leqq b$. $\theta(x)$ is plotted in Figure 10.1.

The points a, x^5, x^7, x^{10}, x^{12} are strong local minima; x^0, x^4, x^6, x^{11}, b are strong local maxima; x^{12} is the global minimum; x^6 is the global maximum; in this problem. At the point x^3 the derivative of $\theta(x)$ is zero, and so it is a stationary point (satisfies the necessary optimality condition $\frac{d\theta(x)}{dx} = 0$) even though it is neither a local minimum or maximum. In each of the intervals $x^1 \leqq x \leqq x^2$, and $x^8 \leqq x \leqq x^9$, $\theta(x)$ is a constant. x^1, x^2 are weak local minima; and x^8, x^9 are weak local maxima. Every point x satisfying $x^1 < x < x^2$, $x^8 < x < x^9$ is both a weak local minimum and a weak local maximum.

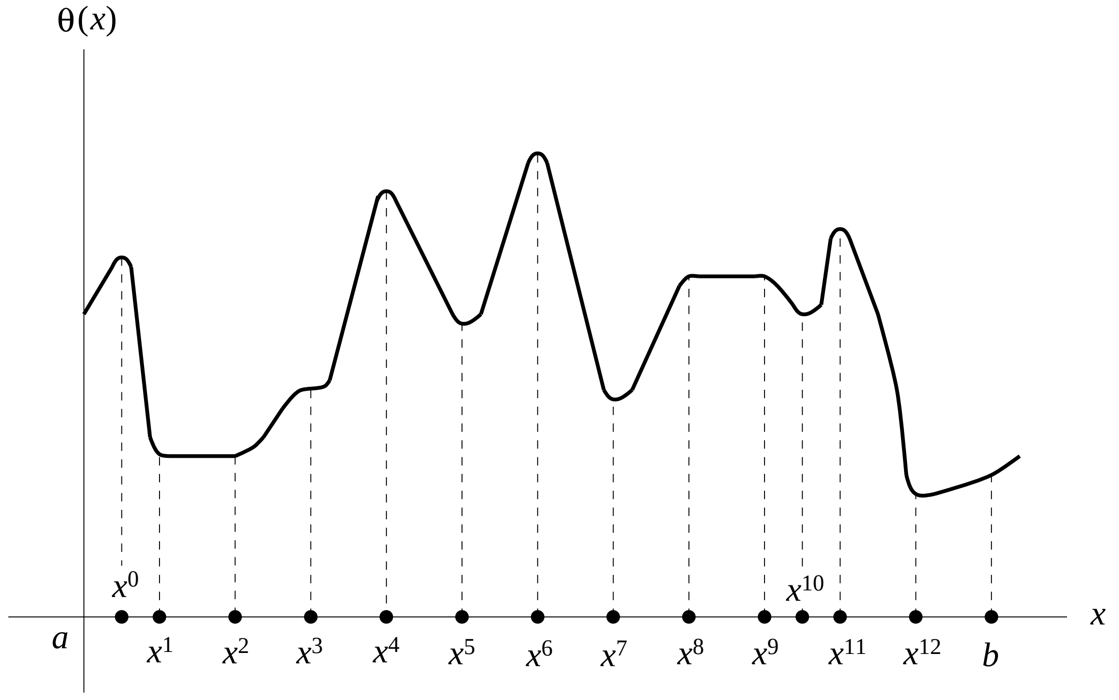

Figure 10.1

10.3 TYPES OF NONLINEAR PROGRAMS, WHAT CAN AND CANNOT BE DONE EFFICIENTLY BY EXISTING METHODS

Every local minimum is a global minimum for the problem of minimizing a convex objective function on a convex set. Likewise, every local maximum is a global maximum for the problem of maximizing a concave function on a convex set. Problems of this type are considered to be nice problems in nonlinear programming, they are called convex programming problems. The other class of NLPs in which a nonconvex objective function is required to be minimized, or in which the set of feasible solutions is not convex, are called nonconvex programming problems.

In general, it is very hard to find the global minimum, or even to check whether a given feasible solution is a global minimum in a nonconvex programming problem. Efforts have been made to find global minima by enumerating all local minima, but these methods tend to be very inefficient. The enormity of this task can be appreciated when we realize that some of the most difficult problems in mathematics that have remained unresolved for centuries, can be posed as nonconvex programming problems.

As an example, consider **Fermat's last Theorem** (unresolved since 1637 AD, see [10.34]) which states that the equation: $x^{\mathbf{n}} + y^{\mathbf{n}} - z^{\mathbf{n}} = 0$, has no solution in integers in the region $x \geqq 1,\ y \geqq 1,\ z \geqq 1,\ n \geqq 3$. Consider the following NLP, where α is some positive parameter, π denotes the irrational number which is the length of the circumference of the circle with unit diameter in $\mathbf{R}^2$, and $\cos\theta$ denotes the cosine function of the angle θ measured in radians.

$$\begin{array}{ll} \text{minimize} & (x^{\mathbf{n}} + y^{\mathbf{n}} - z^{\mathbf{n}})^{\mathbf{2}} + \alpha((-1+\cos(2\pi x))^{\mathbf{2}} + (-1+\cos(2\pi y))^{\mathbf{2}} + \\ & (-1+\cos(2\pi z))^{\mathbf{2}} + (-1+\cos(2\pi n))^{\mathbf{2}}) \\ \text{subject to} & x, y, z \geqq 1,\ n \geqq 3. \end{array} \qquad (10.1)$$

(10.1) is a linearly constrained NLP. It can be verified that Fermat's last Theorem is false iff the optimum objective value in (10.1) is 0 and attained, since any feasible solution (x, y, z, n) to (10.1) which makes the objective value zero provides a counterexample to Fermat's last Theorem. (10.1) is a nonconvex programming problem in which every integer feasible solution is a local minimum. The objective function in (10.1) is a sum of several penalty terms. The number of distinct local minima can be very large even in nonconvex programming problems that do not have such penalty terms in the objective function. As an example, consider the concave minimization problem

$$\begin{array}{ll} \text{minimize} & \theta(x) = -\sum_{j=1}^{n}(x_j - (1/2))^{\mathbf{2}} \\ \text{subject to} & 0 \leqq x_j \leqq 1,\ j = 1 \text{ to } n. \end{array} \qquad (10.2)$$

Each of the $2^{\mathbf{n}}$ extreme points of the set of feasible solutions of (10.2) is a local minimum. Unfortunately, there are no techniques known for determining how many local minima a general nonconvex programming problem has, other than plain enumeration. In nonconvex programming problems, since in general it is very difficult to guarantee that a global minimum will be obtained, the best thing that we can expect from an algorithm is that it leads to a point satisfying a necessary condition for being a local mimimum, and many of the descent type methods discussed in this chapter do that. In these methods, the terminal solution obtained may depend on the initial point with which the method is initiated. Usually, by running the algorithm with different initial points, several local minima may be obtained, and the best among them might be a reasonably good solution for the problem.

Starting the algorithm with an initial point, suppose a local minimum $\overline{x}$ is obtained for a nonconvex programming problem. A technique often used to move to a different local minimum is to add a penalty term like $\alpha/(||x - \overline{x}||)^{\mathbf{p}}$ where $\alpha > 0$ and $p \geqq 2$, to the objective function, and use the algorithm again on the augmented problem. As x approaches $\overline{x}$, the penalty term $\alpha/(||x - \overline{x}||)^{\mathbf{p}}$ blows up to ∞, and this guarantees that the algorithm moves to a point different from $\overline{x}$. But this may not be a satisfactory approach to enumerate the local minima in a nonconvex program, because of the numerical difficulties created by the addition of the penalty terms to avoid previously obtained local minima. Also, the augmented problem may have new local minima which are not local minima of the original problem.

Because of this, if someone can establish the global minimum in a class of nonconvex programming problems, it is considered to be a mathematical breakthrough and becomes a major international headline item. An example of this is the recent breakthrough on establishing the minimum value of the permanent of a doubly stochastic matrix of order n. Given a square matrix $A = (a_{ij})$ of order n, its permanent is defined by

$$f(A) = \sum([(a_{1p_1}) \dots (a_{np_n})] : \text{ sum over all the } n! \text{ permutations } (p_1, \dots, p_n) \text{ of } \{1, \dots, n\}).$$

A doubly stochastic matrix of order n is a nonnegative square matrix $X = (x_{ij})$ of order n, whose row sums and column sums are all equal to 1. The problem of minimizing the permanent of doubly stochastic matrix of order n is therefore the NLP: find a square matrix $X = (x_{ij})$ of order n to

$$\begin{aligned} \text{minimize} \quad & f(X) \\ \text{subject to} \quad & \sum_{j=1}^{n} \quad x_{ij} = 1, \ i = 1 \text{ to } n \\ & \sum_{i=1}^{n} \quad x_{ij} = 1, \ j = 1 \text{ to } n \\ & \qquad x_{ij} \geqq 0, \ i,j = 1 \text{ to } n. \end{aligned}$$

The objective function in this NLP is nonconvex, hence, this is a nonconvex programming problem. In 1926 B. L. vanderWaerden [10.40] conjectured that the global optimum for this problem is the doubly stochastic matrix $(\overline{x}_{ij})$ in which $\overline{x}_{ij} = 1/n$ for all i, j; with an optimum objective value of $n!/n^{\mathbf{n}}$. This conjecture resisted the attacks of many of the world's greatest mathematicians, but was finally resolved in the affirmative by G. P. Egorychev in 1980, see references [10.10, 10.11, 10.20].

10.4 CAN WE AT LEAST COMPUTE A LOCAL MINIMUM EFFICIENTLY?

In convex programming problems, any point satisfying any of the well known necessary optimality conditions such as the KKT conditions, is a local minimum and therefore it is also a global minimum for the problem. To solve a convex programming problem, any algorithm that is guaranteed to find a KKT point, if one exists, is thus adequate. Most of the algorithms for solving NLP's discussed in this book can be shown to converge to a KKT point, if one exists, and so these algorithms compute local, and thus global minima when applied on convex programming problems.

In a nonconvex program, given a feasible solution x satisfying the usual necessary optimality conditions, it may or may not even be a local minimum. If x does not satisfy the sufficient optimality condition given in Appendix 4 for being a local minimum, it may be very hard to verify whether it is a local minimum. As an example, consider the problem discussed in Section 2.9.3

$$\begin{array}{ll} \text{minimize} & x^T D x \\ \text{subject to} & x \geqq 0 \end{array}$$

where D is a given square matrix of order n. When D is not PSD, this NLP is the **simplest nonconvex** NLP.

A sufficient condition for 0 to be a local minimum for this problem is that D be PSD. If D is not PSD, 0 is a local minimum for this problem iff the matrix D is copositive, no efficient methods are known at the moment for doing this. The method discussed in Section 2.9.1 for testing copositiveness is a finite enumeration method, but it may not be practically useful when n is large. As discussed in Section 2.9.3, the problem of checking whether 0 is a local minimum for this problem is a hard problem.

On nonconvex programs involving inequality constraints, existing algorithms can at best guarantee convergence to a KKT point in general. If the KKT point obtained does not satisfy some known sufficient condition for being a local minimum, it is then hard to check whether it is actually a local minimum. However, as mentioned in Section 2.7.6, if the algorithm is based on a descent process (i. e., in a minimization problem, if the algorithm is designed to obtain a sequence of points with decreasing objective values) one can be reasonably confident that the solution obtained is likely to be a local minimum.

10.5 PRECISION IN COMPUTATION

In linear or in convex quadratic programming problems, if all the data are rational numbers, and if an optimum solution exists, there exists an optimum solution which is a rational vector that can be computed exactly with finite precision arithmetic using algorithms like the simplex algorithm or the complementary pivot method discussed earlier. However, in general nonlinear programming, even when the constraints are linear, and all the data in the model is rational, there may be optimum solutions, but no rational optimum solution. For example consider the simple one dimensional optimization problem: find $x \in \mathbf{R}^1$ that minimizes $f(x) = -2x + (x^3/3)$ subject to $x \geqq 0$. The unique optimum solution of this problem is $\overline{x} = \sqrt{2}$, an irrational number, so we can never compute the exact optimum solution of this problem on digital computers that operate with finite precision arithmetic.

Hence, when dealing with general nonlinear programs, emphasis is placed on getting an approximate optimum solution. In practical implementations, nonlinear algorithms are usually terminated when optimality conditions are satisfied to a reasonable degree of approximation, or when it is evident that the algorithm has obtained an interval of sufficiently small length containing the true optimum solution.

10.6 RATES OF CONVERGENCE

The algorithms discussed in this chapter are iterative in nature. They generate a sequence of points $\{x^r : r = 0, 1, 2, \ldots\}$ beginning with an initial point x^0. Under some conditions on the problem being solved, for most of these methods, it is usually possible to prove that the sequence converges in the limit to a point x^* which is a point satisfying the necessary optimality conditions for a local minimum. Even when this convergence is mathematically proven, the method is useful for solving practical problems only if x^r converges rapidly to x^* as r increases. Here we discuss how this rate of convergence is measured mathematically.

Finite Termination Property: The sequence is said to have this property, if there exists a finite value N such that $x^N = x^*$ and the method terminates.

Quadratic Termination Property: The method is said to have this property if the sequence generated terminates in a known finite number of iterations when applied to a strictly convex quadratic function minimization problem.

Suppose the method does not have either of the above properties. Then it generates the truly infinite sequence $\{x^r : r = 0, 1, 2, \ldots\}$. Assume that the sequence converges to x^*, that $x^r \neq x^*$ for any r. The measure of the rate of convergence of this sequence, tries to assess the improvement that occurs in each step, that is, in effect it measures how close x^{r+1} is to x^* compared to the closeness of x^r to x^*, as r goes to ∞. The converging sequence $\{x^r\}$ is said to converge with order k (or to have an asymptotic convergence rate k) if k is the largest number such that $\text{limit}_{r\to\infty}(\|x^{r+1} - x^*\|/\|x^r - x^*\|^{\mathbf{k}}) < \infty$. When $k = 1$, the sequence is said to have **linear** (or **first order**, or **geometric**) convergence rate, if $\text{limit}_{r\to\infty}(\|x^{r+1} - x^*\|/\|x^r - x^*\|) = \gamma < 1$. In this case, the quantity γ is called the convergence ratio of the sequence. If in fact $\gamma = 0$ in this case, the sequence is said to have superlinear convergence rate.

As an example consider the sequence of real numbers $\{\alpha^r : r = 0, 1, \ldots\}$ where $0 < \alpha < 1$. The sequence converges to zero linearly. On the other hand the sequence of real numbers $\{x^r = (1/r) : r = 1, 2, \ldots\}$ converges to zero with $k = 1$, but its rate of convergence is not linear, since $\text{limit}_{r\to\infty}(\|x^{r+1}\|/\|x^r\|) = \text{limit}_{r\to\infty}((r/(r+1)) = 1$ which is not strictly less than one.

If $k = 2$, the sequence $\{x^r\}$ is said to have **quadratic** (or **second order**) **convergence rate**. Quadratic convergence is rapid, since it implies that once the sequence reaches a small neighborhood of x^*, the error in a step decreases as the square of the error in the previous step (i. e. , the number of digits to which x^r agrees with x^* begin to double after each step, after a certain number of steps).

Summary of Later Sections

In the following sections we discuss various descent methods in common use for solving linearly constrained NLPs. These algorithms typically use some unconstrained minimization algorithms and algorithms for solving nonlinear programs in a single variable (the so-called line minimization algorithms) as subroutines. So we survey these algorithms first.

10.7 SURVEY OF SOME LINE MINIMIZATION ALGORITHMS

The line minimization problem is the problem of minimizing a real valued function $f(\lambda)$ of one variable λ, either over the whole real line, or over the half-line $\lambda \geqq l$ for a specified number l, or over a specified finite interval $[l, u] = \{\lambda : l \leqq \lambda \leqq u\}$. Assuming that $f(\lambda)$ is continuously differentiable, the global minimum for $f(\lambda)$ in the interval $l \leqq \lambda \leqq u$ is the point λ^* in this interval which gives the minimum value for $f(\lambda)$ among those λ satisfying $\frac{df(\lambda)}{d\lambda} = 0$, and the points l, u, if these are finite. In fact if $f(\lambda)$ is concave and l, u are finite, the global minimum for $f(\lambda)$ in the interval $l \leqq \lambda \leqq u$ is either l or u, whichever gives a smaller value for $f(\lambda)$. See Figure 10.2.

In the interval $[a, b]$ if $f'(a) > 0$, a is a local minimum for $f(\lambda)$; and if $f'(b) < 0$, b is a local minimum for $f(\lambda)$.

When $f(\lambda)$ is a general function, a **bracket** is defined to be an interval in the feasible region which contains the minimum. When the derivative $f'(\lambda) = \frac{df(\lambda)}{d\lambda}$ is not available, a bracket usually refers to an interval $[\lambda_1, \lambda_3]$ in the feasible region, satisfying the property that we have a λ_2 satisfying $\lambda_1 < \lambda_2 < \lambda_3$ and $f(\lambda_2) \leqq$ minimum $\{f(\lambda_1), f(\lambda_3)\}$. If the derivative $f(\lambda)$ is available, a bracket usually refers to an interval $[\lambda_1, \lambda_2]$ with $\lambda_1 < \lambda_2$, satisfying the property that $f'(\lambda_1) < 0$ and $f'(\lambda_2) > 0$.

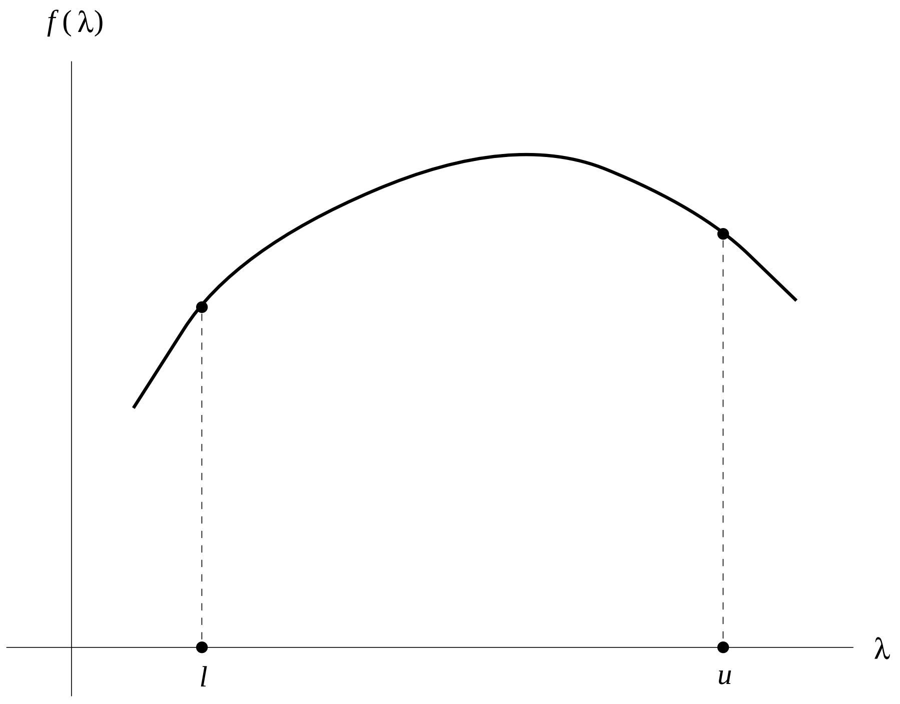

Figure 10.2 The global minimum for one dimensional concave minimization problem is a boundary point (l here).

How to Select an Initial Bracket?

First consider the problem in which we are required to minimize $f(\lambda)$ over the entire real line. Begin with an initial point λ_0 and choose a positive step lenth Δ. Compute $f(\lambda_0)$ and $f(\lambda_1)$, where $\lambda_1 = \lambda_0 + \Delta$. If $f(\lambda_1) < f(\lambda_0)$, the direction of increasing λ is the right direction to pursue; otherwise, replace Δ by $-\Delta$ to reverse the direction and go through the procedure discussed next. Define $\lambda_r = \lambda_{r-1} + 2^{\mathbf{r-1}}\Delta$ for $r = 2, 3, \dots$ as long as they keep on decreasing, until either the upper bound on λ is reached or a value k for r is found such that $f(\lambda_{k+1}) > f(\lambda_k)$. In this case we have λ_{k-1}, λ_k, λ_{k+1} satisfying $f(\lambda_k) < f(\lambda_{k-1})$, $f(\lambda_{k+1}) > f(\lambda_k)$. Among the four points λ_{k-1}, λ_k, $(\lambda_k + \lambda_{k+1})/2$, and λ_{k+1}, drop either λ_{k-1} or λ_{k+1}, whichever is farther from the point in the pair $\{\lambda_k, (\lambda_k + \lambda_{k+1})/2\}$ that yields the smallest value to $f(\lambda)$. Let the remaining points be called λ_a, λ_b, λ_c, where $\lambda_a < \lambda_b < \lambda_c$. These points are equi-distant, and $f(\lambda_b) \leqq f(\lambda_c)$, $f(\lambda_b) \leqq f(\lambda_a)$. So this interval λ_a to λ_c brackets the minimum.

If the problem is to minimize $f(\lambda)$ over $\lambda \geqq l$ or $u \geqq \lambda \geqq l$, it is reasonable to expect that $f(\lambda)$ decreases as λ increases through l (i. e., the derivative $f'(l) < 0$, otherwise l is itself a local minimum for the problem). So in these problems, we can get a bracket by beginning with $\lambda_0 = l$ and applying the above procedure.

10.7.1 The Golden Section Search Method

The function $f(\lambda)$ is said to be a unimodal function in the interval $a \leqq \lambda \leqq b$ if it has a unique local minimum in the interval. See Figures 10.3, 10.4.

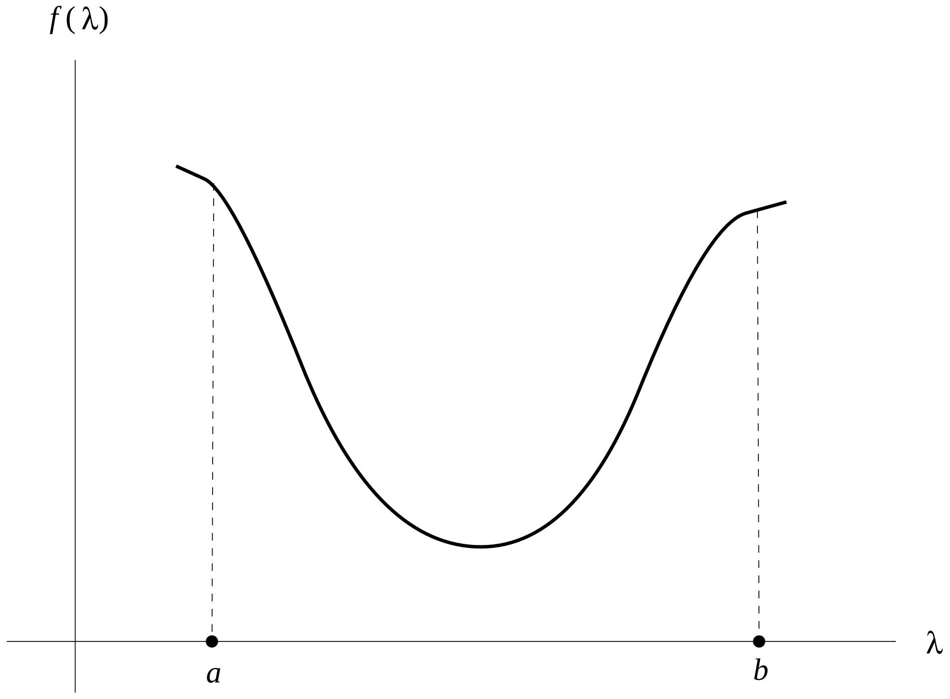

Figure 10.3 A unimodal function in the interval $[a, b]$.

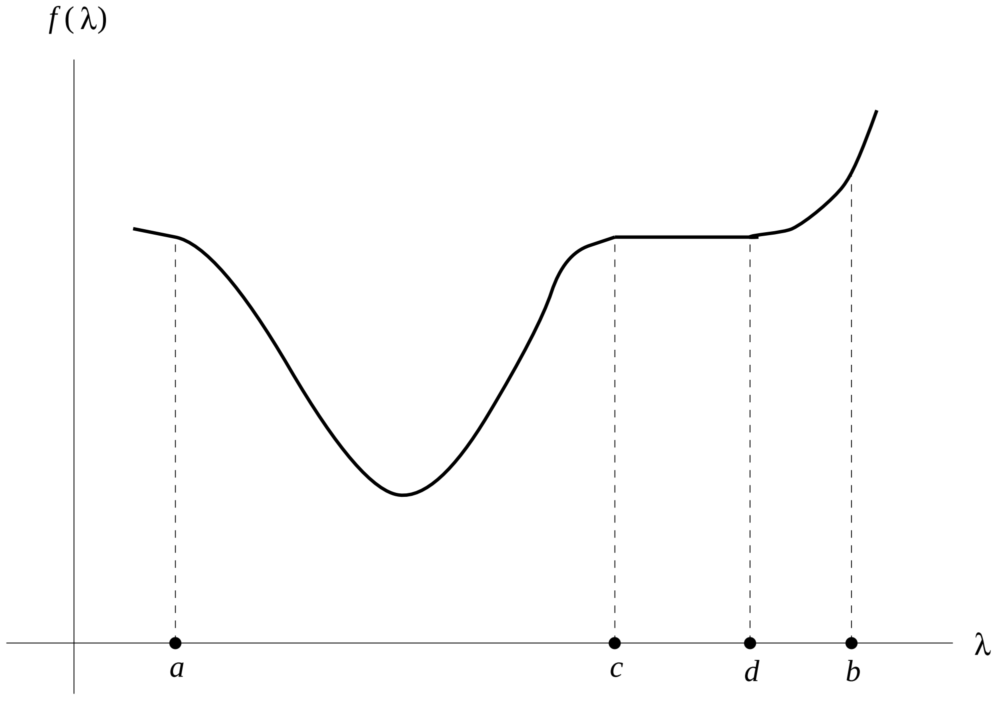

Figure 10.4 This function is constant in the interval $c < \lambda < d$, so every point in this interval is a local minimum. So this function is not unimodal in the interval $[a, b]$.

In many practical applications, it is reasonable to assume that the interval has been narrowed down using prior knowledge of the problem such that the objective function has a single minimum in the interval. A unimodal function $f(\lambda)$ in the interval $a \leqq \lambda \leqq b$ satisfies the property that there exists a unique λ_* in the interval (this λ_* is the minimum) such that given any λ_1, λ_2 in the interval with $\lambda_1 < \lambda_2$, if $\lambda_2 < \lambda_*$ we have $f(\lambda_1) > f(\lambda_2)$; and if $\lambda_* < \lambda_1$, we have $f(\lambda_1) < f(\lambda_2)$. The golden section search method is a method for minimizing a unimodal function in an interval by sectioning (i. e., interval reduction) using only function values evaluated at selected points.

The number $\tau \equiv 2/(1+\sqrt{5}) \simeq .618$ is known as the golden ratio. Let $[\alpha, \beta]$ be the current interval in which the minimum is known to lie. If function value has not been evaluated at any interior point in this interval, let $\lambda_1 = \alpha + .382(\beta - \alpha)$, $\lambda_2 = \alpha + .618(\beta - \alpha)$, evaluate $f(\lambda_1)$, $f(\lambda_2)$ (depending on what happened in the previous step, it is possible that the function value at one of these points λ_1 or λ_2 has already been computed in the previous steps). If $f(\lambda_1) < f(\lambda_2)$, the minimum is contained in the interval $[\alpha, \lambda_2]$. If $f(\lambda_1) > f(\lambda_2)$, the minimum is contained in the interval $[\lambda_1, \beta]$. If $f(\lambda_1) = f(\lambda_2)$, the minimum is contained in the interval $[\lambda_1, \lambda_2]$. Repeat this process with the new interval.

There is a reduction in the length of the interval of uncertainty (i. e., the bracket length) by a factor of .618 or more in each step. The length of the interval of uncertainty converges linearly to zero. When the length of the interval of uncertainty has become less than a specified tolerance, ε, any point in the final interval could be taken as an approximation for the minimum.

10.7.2 The Method of Bisection

This method can be used if $f(\lambda)$ is continuously differentiable and the derivative $f'(\lambda)$ can be computed. It starts with an initial bracket for the minimum $[a, b]$ satisfying $f'(a) < 0$ and $f'(b) > 0$. Evaluate $f'((a+b)/2)$. If $f'((a+b)/2) = 0$, the point $(a+b)/2$ satisfies the first order necessary condition for a local minimum. If $f'((a+b)/2) > 0$, take $[a, (a+b)/2]$ as the new bracket and continue. If $f'((a+b)/2) < 0$, take $[(a+b)/2, b]$ as the new bracket and continue.

Since the bracket is cut in half each time, the length of this interval converges to zero linearly. When its length has become less than a specified tolerance ε, any point in the final interval could be taken as an approximation to the minimum. One disadvantage of this method is that it relies totally on the values of the derivative $f'(\lambda)$ and does not use the values of the function $f(\lambda)$ being minimized.

10.7.3 Newton's Method

This is a second order gradient method that can be used if $f(\lambda)$ is twice continuously differentiable and the second derivative $f''(\lambda)$ can be computed easily either through a subroutine or by using a finite difference approximation, and $f(\lambda)$ is required to be minimized over the entire real line. The method is the application of the Newton-Raphson method to find a solution of the equation: $f'(\lambda) = 0$. The method generates a sequence $\{\lambda_r : r = 0, 1, \dots\}$ beginning with an initial point λ_0. Given λ_r, the second order Taylor series approximation for $f(\lambda)$ at λ_r is $f(\lambda_r) + f'(\lambda_r)(\lambda - \lambda_r) + (1/2)f''(\lambda_r)(\lambda - \lambda_r)^2$. If $f''(\lambda_r) > 0$, this has a minimum at

$$\lambda_{r+1} = \lambda_r - f'(\lambda_r)/f''(\lambda_r). \tag{10.3}$$

Equation (10.3) gives the iterative scheme for Newton's method. The method is not suitable to be used if $f''(\lambda)$ turns out to be $\leqq 0$ at any point encountered during the algorithm. It is quite suitable if an initial point λ_0 in the vincinity of a local minimum is known. In the vincinity of a minimum, the second derivative $f''(\lambda)$ is of constant sign (nonnegative) and the first derivative $f'(\lambda)$ changes sign from a negative to a positive value. If $f(\lambda)$ is a quadratic function with a minimum, this method finds the minimum in one step. In general, any twice continuously differentiable function has a Taylor series expansion around a point, the first three terms of this series (which form a quadratic function) are dominant when the point is in the vincinity of the minimum. The method has rapid convergence (quadratically) once the vincinity of the minimum is reached. A result on the convergence rate of this method follows as a corollary of Theorem 10.1, where a convergence rate result for Newton's method applied to find the unconstrained minimum of a real valued function $\theta(x)$ over $x \in \mathbf{R}^n$ is proved. See references [10.9, 10.13, 10.26, 10.33] for results on the convergence and rates of convergence of Newton's method.

10.7.4 Modified Newton's Method

Several modifications have been proposed for Newton's method to handle cases where a good initial point is not available to initiate Newton's method, or when a point satisfying $f''(\lambda) \leqq 0$ is encountered during the method, and to handle the problem in which the feasible range is a specified interval and not the entire real line. We discuss one such modification here. We consider the problem of minimizing a twice continuously differentiable function $f(\lambda)$ in the interval $[a, c] = \{\lambda : a \leqq \lambda \leqq c\}$ and we have a piont b satisfying $a < b < c$ and $f(b) <$ minimum $\{f(a), f(c)\}$. This method generates a sequence of points $\{\lambda_r : r = 0, 1, \dots\}$ satisfying the property that the entire sequence lies in the interval $[a, c]$ and that $f(\lambda_{r+1}) < f(\lambda_r)$ for all r. Initiate the method with $\lambda_0 = b$, and select a constant α satisfying $0 < \alpha < 1$. The quantity α is called the **attenuation factor**.

Given λ_r, the point obtained by moving in the direction of $f'(\lambda_r)$ a step of length β is $\lambda_r - \beta f'(\lambda_r)$. From the Taylor series, $f(\lambda_r - \beta f'(\lambda_r)) = f(\lambda_r) - \beta(f'(\lambda_r))^2 +$ error term, where the error term tends to zero faster than β. So, if $\beta > 0$, we make improvement in the objective value by this move. Notice that Newton's method takes

$\beta = 1/f''(\lambda_r)$ to get the next point in the sequence. In this method you do the following.

(a) If $f''(\lambda_r) > 0$ compute $y_r = \lambda_r - f'(\lambda_r)/f''(\lambda_r)$. If $y_r \in [a, c]$ and $f(\lambda_r) - f(y_r) \geqq (\alpha/2)(f'(\lambda_r))^2/f''(\lambda_r)$, define $\lambda_{r+1} = y_r$. If $y_r \in [a, c]$ but $f(\lambda_r) - f(y_r) < (\alpha/2)(f'(\lambda_r))^2/f''(\lambda_r)$, use a first order Armijo step size procedure which requires the determination of the smallest nonnegative integer s satisfying

$$(\lambda_r - f'(\lambda_r)/2^s) \in [a, c], \text{ and}$$
$$f(\lambda_r) - f(\lambda_r - f'(\lambda_r)/2^s) > (\alpha/2^s)(f'(\lambda_r))^2$$

and then define $\lambda_{r+1} = \lambda_r - f'(\lambda_r)/2^s$. The motivation for this step size procedure is explained in Section 10.8.1.

(b) If $f''(\lambda_r) \leqq 0$, define $\delta = -1$ if $f'(\lambda_r) \geqq 0$, $+1$ if $f'(\lambda_r) < 0$ and use the second order Armijo step size procedure. This requires the determination of the smallest nonnegative integer s satisfying

$$(\lambda_r - (f'(\lambda_r)/2^s) + (\delta/2^{s/2})) \in [a, c], \text{ and}$$
$$f(\lambda_r) - f(\lambda_r - (f'(\lambda_r)/2^s) + \delta/2^{s/2}) \geqq \alpha(((f'(\lambda_r))^2/2^s) - f''(\lambda_r)/2^{s+1}).$$

For a finite s satisfying these conditions to exist, it is sufficient that $f''(\lambda_r) < 0$ if $f'(\lambda_r) = 0$. Then define $\lambda_{r+1} = \lambda_r - (f'(\lambda_r)/2^s) + \delta/2^{s/2}$.

Under certain conditions it can be shown that this method has second-order convergence. See references [10.1, 10.26, 10.27].

10.7.5 Secant Method

In Newton's method or modified Newton's method discussed above, we need to compute the value of the second derivative $f''(\lambda_r)$. This may be hard. In the secant method we replace $f''(\lambda_r)$ by its finite difference approximation $(f'(\lambda_r) - f'(\lambda_{r-1}))/(\lambda_r - \lambda_{r-1})$. This is the only change in the Secant method from Newton's or modified Newton's method. The secant method is initiated with two initial points λ_0, λ_1 in the feasible region satisfying $\lambda_0 < \lambda_1$ and $f'(\lambda_0) < 0$, $f'(\lambda_1) > 0$.

10.7.6 The Method of False Position

In the secant method we always use $f'(\lambda_r)$ and $f'(\lambda_{r-1})$ to get a finite difference approximation for $f''(\lambda_r)$ for each r. Even though initially $f'(\lambda_0)$, $f'(\lambda_1)$ are of opposite signs, after some steps it may happen that $f'(\lambda_r)$ and $f'(\lambda_{r-1})$ have the same sign, and this could make the iterates diverge when minimizing over the real line. In this method we make sure that $f''(\lambda)$ is always approximated using the values of $f'(\lambda)$ of opposite signs at two different values of λ. For some r, suppose $f''(\lambda_r)$ was approximated using $f'(\lambda_r)$ and $f'(\lambda_s)$ for an $s \leqq r - 1$. Compute λ_{r+1} using this approximation as under the secant method, and compute $f'(\lambda_{r+1})$. Determine which of $f'(\lambda_t)$ for $t = r$ or s has a sign opposite to that of $f'(\lambda_{r+1})$. Then approximate $f''(\lambda_{r+1})$ by $f'(\lambda_{r+1} - f'(\lambda_t))/(\lambda_{r+1} - \lambda_t)$, and continue in the same way.

10.7.7 Univariate Minimization by Polynomial Approximation Methods

The essential feature of these methods is to approximate the original function $f(\lambda)$ by a simpler function $P(\lambda)$ (normally a second or third degree polynomial) by curve fitting, and then using the minimum of $P(\lambda)$ to approximate that of $f(\lambda)$. These methods are also called polynomial interpolation methods. If the minimum is known to lie in a small enough interval, the application of these methods usually produces very satisfactory results.

Quadratic Interpolation

This method needs an interval of the form $\lambda_1 < \lambda_2 < \lambda_3$ with $f(\lambda_2) < \min\{f(\lambda_1), f(\lambda_3)\}$, a bracket for the minimum, as discussed earlier. λ_2, the initial best point, is the initial point in the sequence. It constructs a quadratic approximation $P(\lambda) = a\lambda^{\mathbf{2}} + b\lambda + c$ which coincides with $f(\lambda)$ at $\lambda = \lambda_1, \lambda_2, \lambda_3$. By the properties mentioned above, $P(\lambda)$ determines a parabola. The three independent pieces of information (value of $P(\lambda)$ = value of $f(\lambda)$ at $\lambda = \lambda_1, \lambda_2, \lambda_3$) are used to determine a, b, c in $P(\lambda)$ uniquely. Since $P(\lambda)$ is a parabola (by the condition imposed), the minimum of $P(\lambda)$ lies in the interval $[\lambda_1, \lambda_3]$ at the point λ satisfying $\frac{dP(\lambda)}{d\lambda} = 0$. It can be verified that this point is

$$\lambda_* = \frac{(\lambda_2^{\mathbf{2}} - \lambda_3^{\mathbf{2}})f(\lambda_1) + (\lambda_3^{\mathbf{2}} - \lambda_1^{\mathbf{2}})f(\lambda_2) + (\lambda_1^{\mathbf{2}} - \lambda_2^{\mathbf{2}})f(\lambda_3)}{2[(\lambda_2 - \lambda_3)f(\lambda_1) + (\lambda_3 - \lambda_1)f(\lambda_2) + (\lambda_1 - \lambda_2)f(\lambda_3)]}$$

λ_* is a minimum for $P(\lambda)$ if

$$\frac{(\lambda_2 - \lambda_3)f(\lambda_1) + (\lambda_3 - \lambda_1)f(\lambda_2) + (\lambda_1 - \lambda_2)f(\lambda_3)}{(\lambda_1 - \lambda_2)(\lambda_2 - \lambda_3)(\lambda_3 - \lambda_1)} < 0$$

a condition which will hold because of the properties satisfied by λ_1, λ_2, λ_3.

It is possible for λ_* to be equal to λ_2 even though this point is far away from a local minimum of $f(\lambda)$. See Figure 10.5. If this happens, the quadratic interpolation has failed to generate a new trial point.

If $|\lambda_* - \lambda_2|$ is not too small, we can replace one of the points in λ_1, λ_2, λ_3 by λ_* so that the new set of three points again satisfies the conditions for a bracket for the minimum of $f(\lambda)$. The best point among these three is the next point in the sequence, and the procedure is repeated with the new bracket. If λ_* and λ_2 are too close (even if they are not equal) repeating the procedure with such close values could lead to numerical problems in the next step. In this case, we select a small distance δ, and take the new point to be either $\lambda_* + \delta$ or $\lambda_* - \delta$ whichever leads to the smallest length new bracket.

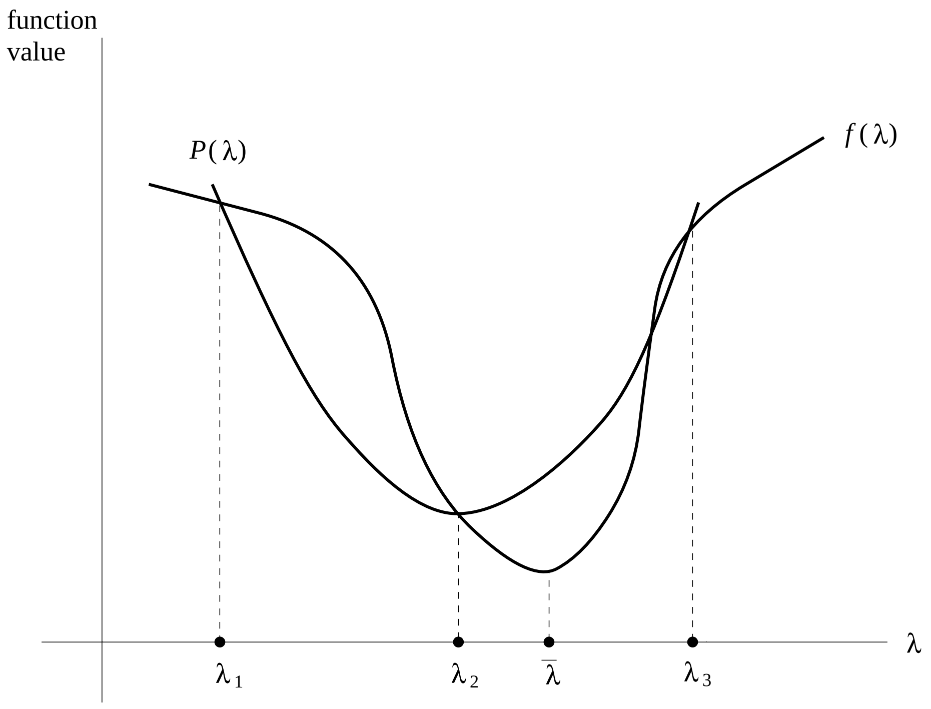

Figure 10.5 The minimum of $f(\lambda)$ in the bracket $[\lambda_1, \lambda_3]$ is at $\bar{\lambda}$. But the minimum of Quadratic approximation, λ_* is the same as λ_2.

Note 10.1 Newton's method is a quadratic approximation method. Given the current point λ_r at which the second derivative $f''(\lambda_r) > 0$, Newton's method constructs a quadratic function $P(\lambda)$ satisfying the three properties $P(\lambda_r) = f(\lambda_r)$, $P'(\lambda_r) = f'(\lambda_r)$ and $P''(\lambda_r) = f''(\lambda_r)$. It can be verified that the function $P(\lambda)$ is just the second order Taylor series approximation to $f(\lambda)$ around this point λ_r, and that the next point in the sequence λ_{r+1} is the minimum of this quadratic approximation $P(\lambda)$.

Cubic Interpolation Method

This method can be used when $f(\lambda)$ is differentiable and the derivative $f'(\lambda)$ can be computed either numerically using a finite difference approximation or computed directly using a subroutine for evaluating it. The method needs a bracket $[\lambda_1, \lambda_2]$ satisfying the property that $f'(\lambda_1) < 0$, $f'(\lambda_2) > 0$. A cubic function $P_3(\lambda) = a\lambda^{\mathbf{3}} + b\lambda^{\mathbf{2}} + c\lambda + d$ can be fitted such that it agrees in value with $f(\lambda)$ at λ_1 and λ_2 and its derivative has the same value as $f'(\lambda)$ at λ_1 and λ_2. From the bracket conditions the minimum of this cubic function occurs inside the bracket at the point λ_* satisfying

$\frac{d}{d\lambda}(P_3(\lambda)) = 0$. It can be verified that

$$\lambda_* = \lambda_1 + (\lambda_2 - \lambda_1)\Big(1 - \frac{f'(\lambda_2) + \nu - \eta}{f'(\lambda_2) - f'(\lambda_1) + 2\nu}\Big)$$

where

$$\begin{aligned}\eta &= \frac{3(f(\lambda_1) - f(\lambda_2))}{\lambda_2 - \lambda_1} + f'(\lambda_1) + f'(\lambda_2)\\ \nu &= (\eta^2 - f'(\lambda_1)f'(\lambda_2))^{1/2}.\end{aligned}$$

If $|f'(\lambda_*)|$ is small, λ_* can be accepted as a good approximation for the minimum. Otherwise, if $f'(\lambda_*) > 0$, repeat the process with $[\lambda_1, \lambda_*]$ as the new bracket. If $f'(\lambda_*) < 0$, repeat the process with $[\lambda_*, \lambda_2]$ as the new bracket.

It can be shown that these polynomial approximation methods have superlinear or better convergence rate under certain conditions. See [10.13, 10.17, A8]. It is possible to develop algorithms based on a combination of sectioning and polynomial interpolation steps.

Difficulty in Computing Derivatives During Line Minimization Steps Encountered in Solving NLPs Involving Several Variables

Let $\theta(x)$ be a continuously differentiable real valued function defined on $\mathbf{R}^n$. Consider the NLP in which $\theta(x)$ is to be minimized, possibly subject to some constraints. Many algorithms for solving such a problem make repeated use of line minimization algorithms to solve problems of the form: given a point $x^0 \in \mathbf{R}^n$ and a search direction $y \in \mathbf{R}^n$, $y \neq 0$, find the step length λ that minimizes $\theta(x^0 + \lambda y)$ subject to $\lambda \geqq 0$.

In this problem, since x^0 and y are given vectors, $\theta(x^0 + \lambda y) = f(\lambda)$ is purely a function of the step length parameter λ. If the problem of minimizing $f(\lambda)$ in $\lambda \geqq 0$ needs the derivative $f'(\lambda)$ for some given value of λ, we use

$$f'(\lambda) = \frac{d}{d\lambda}(\theta(x^0 + \lambda y)) = (\nabla\theta(x^0 + \lambda y))y$$

where $\nabla\theta(x^0 + \lambda y)$ is the row vector of partial derivatives of $\theta(x)$ evaluated at $x = x^0 + \lambda y$. So, the computation of $f'(\lambda)$ needs the evaluation of each of the partial derivatives of $\theta(x)$ at the point $x^0 + \lambda y$, which in the worst case takes n function evaluations (the work would be less if, for example, we know from the structure of $\theta(x)$ that some of these partial derivatives are zero). Thus, evaluating $f(\lambda) = \theta(x^0 + \lambda y)$ needs only one function evaluation; while evaluating $f'(\lambda)$ needs n function evaluations, considerably more work. In the same manner, evaluation of the second derivative $f''(\lambda)$ for any λ, needs n^2 function evaluations in the worst case. These facts should be considered in choosing an algorithm for line minimization, to be used as a subroutine in algorithms for NLPs involving many variables. Since evaluating derivatives ($f'(\lambda)$ or $f''(\lambda)$) requires a lot more function evaluations, typically line minimization algorithms based on function values only, are to be preferred as far as possible.

When $f(\overline{\lambda}) = \theta(x^0 + \overline{\lambda} y)$, the formula $f'(\overline{\lambda}) = (\nabla\theta(x^0 + \overline{\lambda} y))y$ is an analytical formula for the exact derivative of $f(\lambda)$ at $\overline{\lambda}$, and the value of $f'(\overline{\lambda})$ computed using this formula is known as the analytically computed derivative. Since the analytical computation of the derivative is so expensive, it may be appropriate to use an approximation for it. Let ε be a small positive number, it is called the finite difference interval. Then $f'(\overline{\lambda})$ can be approximated by any of the three following quantities

$$\begin{aligned} &\frac{f(\overline{\lambda}) - f(\overline{\lambda} - \varepsilon)}{\varepsilon} \\ \text{or} \quad &\frac{f(\overline{\lambda} + \varepsilon) - f(\overline{\lambda})}{\varepsilon} \\ \text{or} \quad &\frac{f(\overline{\lambda} + \varepsilon) - f(\overline{\lambda} - \varepsilon)}{2\varepsilon}. \end{aligned}$$

The topmost quantity is called the backward-difference approximation, the middle quantity is known as the forward-difference approximation, and the bottom quantity is known as the central-difference approximation, to $f'(\overline{\lambda})$. If the value of $f(\overline{\lambda})$ is already known, the computation of the forward or backward-difference approximation to $f'(\overline{\lambda})$ needs one more function evaluation, whereas the computation of the central-difference approximation needs two more function evaluations. If ε is small compared to $|f'(\overline{\lambda})|$ and the magnitude of $|f''(\lambda)|$ in the neighborhood of $\overline{\lambda}$, the error in approximation will be small, because $f(\overline{\lambda} + \varepsilon) = f(\overline{\lambda}) + \varepsilon f' + \frac{\varepsilon^2}{2} f''(\overline{\lambda} + \gamma)$ for some $0 \leqq \gamma \leqq \varepsilon$, by Taylor's theorem. Thus with a suitable choice of the finite difference interval, these finite difference approximations provide a reasonable approximation to the derivative, with much less computational effort than that involved in using the analytical formula. Because of this, many professional software packages for NLP algorithms use finite difference approximations to the derivatives.

Even the partial derivatives of $\theta(x)$ can be approximated by finite difference approximations. Let I be the unit matrix of order n. Then

$$\begin{aligned} &\frac{\theta(x) - \theta(x - \varepsilon I_{.j})}{\varepsilon} \\ \text{or} \quad &\frac{\theta(x + \varepsilon I_{.j}) - \theta(x)}{\varepsilon} \\ \text{or} \quad &\frac{\theta(x + \varepsilon I_{.j}) - \theta(x - \varepsilon I_{.j})}{2\varepsilon} \end{aligned}$$

where ε is the suitable finite difference interval, are the backward, forward and central-difference approximations for the partial derivative $\frac{\partial\theta(x)}{\partial x_j}$, respectively.

10.7.8 Practical Termination Conditions for Line Minimization Algorithms

In practice, line minimization algorithms discussed above are terminated either when the bracket length is small, or when a point λ satisfying $|f'(\lambda)| < \varepsilon$ for some specified tolerance ε is obtained, or when the improvement in objective value between two consecutive points obtained in the method is small, or when the difference between two consecutive points obtained under the method is small. At termination, if we have a bracket for the minimum, a final interpolation step can be carried out to provide the approximate location of the minimum in the bracket.

10.7.9 Line Minimization Algorithms Based on Piecewise Linear and Quadratic Approximations

In this section we discuss new line minimization algorithms based upon a combination of piecewise linear (or polyhedral) and quadratic approximations, due to C. Lemarechal and R. Mifflin [10.23, 10.28, 10.29]. These algorithms are rapidly convergent, and seem best suited as line search subroutines in higher dimensional optimization algorithms.

Let $f(\lambda) : \mathbf{R}^1 \to \mathbf{R}^1$ be the real valued function defined on $\mathbf{R}^1$ which is required to be minimized over $\lambda \in \mathbf{R}^1$. At any given λ, the limit (if it exists) of $\frac{f(\lambda+\varepsilon)-f(\lambda)}{\varepsilon}$ as $\varepsilon \to 0$ through positive values is known as the right derivative of $f(\lambda)$ at λ and denoted by $f'_+(\lambda)$, the limit of the same quantity as $\varepsilon \to 0$ through negative values is known as the left derivative of $f(\lambda)$ at λ and is denoted by $f'_-(\lambda)$. If $f(\lambda)$ is differentiable at λ, then $f'_-(\lambda) = f'_+(\lambda) = f'(\lambda)$. If $f(\lambda)$ is convex, these $f'_-(\lambda)$ and $f'_+(\lambda)$ exist and they satisfy

$$\text{if } \lambda < \gamma, \text{ then } f'_-(\lambda) \leqq f'_+(\lambda) \leqq f'_-(\gamma) \leqq f'_+(\gamma).$$

When $f(\lambda)$ is convex, the subdifferential $\partial f(\lambda)$ is the line segment $[f'_-(\lambda), f'_+(\lambda)]$, and a necessary and sufficient condition for a λ_* to be the minimizer point for $f(\lambda)$ is:

$$f'_-(\lambda_*) \leqq 0 \leqq f'_+(\lambda_*).$$

For the moment, let $g(\lambda)$ denote the derivative $f'(\lambda)$ if it exists; or a number from $\partial f(\lambda)$, that is, a subgradient of $f(\lambda)$ at λ, otherwise. Given two points $\overline{\lambda}$ and $\overline{\gamma}$ satisfying the properties that $f(\overline{\lambda}) \leqq f(\overline{\gamma})$ and $g(\overline{\lambda})g(\overline{\gamma}) < 0$, the interval between $\overline{\lambda}$ and $\overline{\gamma}$ is a bracket for the minimum.

Polyhedral Approximation

The affine functions $f(\overline{\lambda}) + g(\overline{\lambda})(\lambda - \overline{\lambda})$, $f(\overline{\gamma}) + g(\overline{\gamma})(\lambda - \overline{\gamma})$, are the linearizations of $f(\lambda)$ at $\lambda = \overline{\lambda}, \overline{\gamma}$ respectively. The pointwise supremum function $P(\lambda) = \max.\{f(\overline{\lambda}) + g(\overline{\lambda})(\lambda - \overline{\lambda}), f(\overline{\gamma}) + g(\overline{\gamma})(\lambda - \overline{\gamma})\}$ provides a piecewise linear or polyhedral approximation for $f(\lambda)$ in the interval between $\overline{\lambda}$ and $\overline{\gamma}$. If $f(\lambda)$ is convex, this piecewise linear function underestimates $f(\lambda)$ at each point in the interval, see Figure 10.6. The point where this piecewise linear function attains its minimum is the point that equalizes the two expressions inside the max., it is $\overline{\lambda}+d^P$, where

$$d^P = \frac{f(\overline{\lambda}) - f(\overline{\gamma}) - g(\overline{\gamma})(\overline{\lambda}-\overline{\gamma})}{g(\overline{\gamma}) - g(\overline{\lambda})}.$$

This d^P provides the polyhedral approximation step from the point $\overline{\lambda}$ for the line minimization problem. If $f(\lambda)$ is convex, the numerator in d^P is $\geqq 0$ and $\overline{\lambda}+d^P$ lies in the interval between $\overline{\lambda}$ and $\overline{\gamma}$.

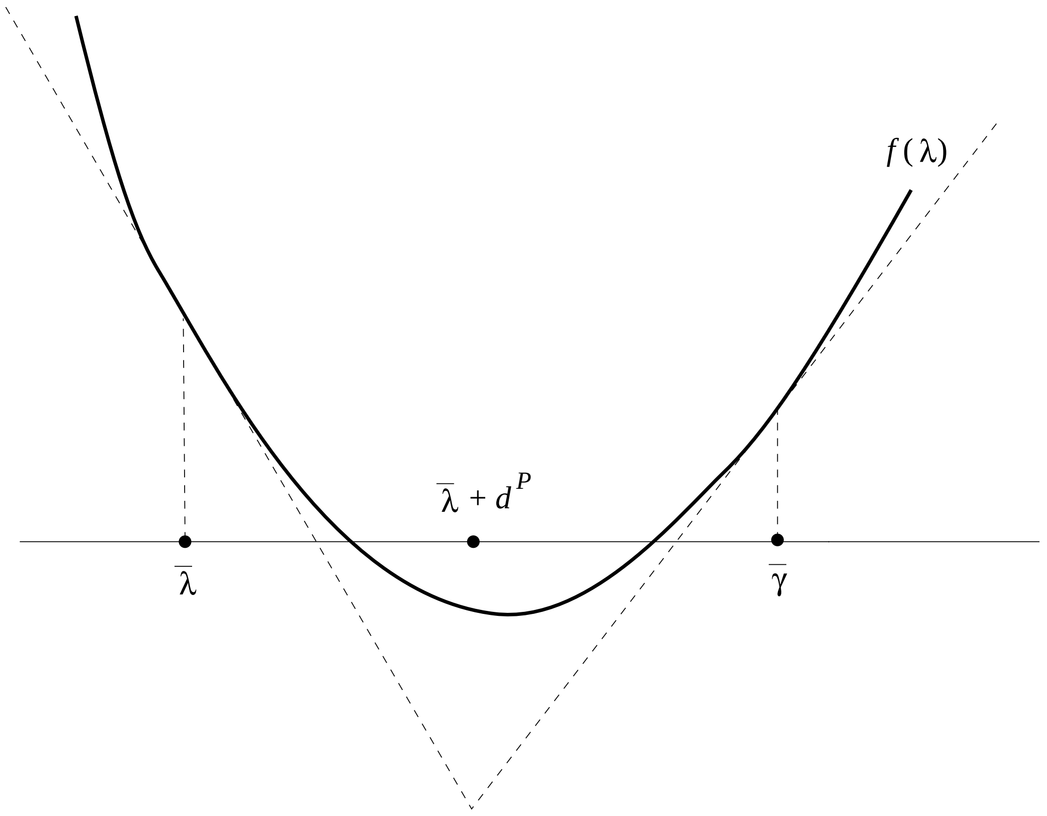

Figure 10.6 A Polyhedral approximation (the dashed lines) for $f(\lambda)$, and the point $\overline{\lambda}+d^P$ where it attains its minimum.

Quadratic Approximation

A quadratic approximation for $f(\lambda)$ at $\lambda=\overline{\lambda}$ is of the form

$$Q(\lambda) = f(\overline{\lambda}) + g(\overline{\lambda})(\lambda-\overline{\lambda}) + \frac{1}{2}(\lambda-\overline{\lambda})^2 G(\overline{\lambda})$$

where $G(\overline{\lambda})$ approximates the second derivative of $f(\lambda)$ and is determined in a one-sided secant manner, that is,

$$G(\overline{\lambda}) = \frac{g(\overline{\lambda}) - g(t)}{\overline{\lambda} - t}$$

where t is a point such that $\overline{\lambda}$ is in the interval between t and the minimizer of $f(\lambda)$. If $f(\lambda)$ is convex, $G(\lambda)$ is ≥ 0. If $G(\overline{\lambda}) > 0$, the minimum of $Q(\lambda)$ is attained at $\overline{\lambda} + d^Q$ where

$$d^Q = \frac{-g(\overline{\lambda})}{G(\overline{\lambda})}.$$

If $G(\lambda) \leqq 0$, $|d^Q| = +\infty$. d^Q is the quadratic approximation step from $\overline{\lambda}$ for the line minimization problem.

The algorithm uses a step that is the shorter of the quadratic approximation and the polyhedral approximation steps. Some modifications are made to these steps if the functions are not convex, to guarantee convergence to at least a stationary point.

These methods generate two sequences $\{\lambda_r\}$, $\{\gamma_r\}$ where for each r, λ_r and γ_r are on opposite sides of the minimizing point λ_*. The sequence $\{f(\lambda_r)\}$ will be non-increasing, and $|\lambda_r - \gamma_r|$ is a decreasing sequence, since at least one of the two points λ_r, γ_r changes in each step.

We describe different versions of the algorithm in various numbered subsections in the following, for ease of cross referencing.

1 Line Minimization of a Convex Function

Assume that $f(\lambda)$ is convex and that it is required to find the point λ_* that minimizes $f(\lambda)$ over $\lambda \in \mathbf{R}^1$. In this subsection, $g(\lambda)$ denotes $f'(\lambda)$ if $f(\lambda)$ is differentiable at λ, or a subgradient of $f(\lambda)$ at λ otherwise (i. e., a point from $\partial f(\lambda)$, the interval between $f'_-(\lambda)$ and $f'_+(\lambda)$). The method initially needs two points λ_1 and γ_1 satisfying

$$f(\lambda_1) \leqq f(\gamma_1) \text{ and } g(\lambda_1)g(\gamma_1) < 0.$$

A pair of points like this can be generated by some initialization procedure. In this case λ_* is in the interval between λ_1 and γ_1. Choose $G(\lambda_1) = (g(\gamma_1) - g(\lambda_1))/(\gamma_1 - \lambda_1)$. We will now describe the general step.

Step r. At the beginning of this step we have λ_r, γ_r satisfying

$$f(\lambda_r) \leqq f(\gamma_r) \text{ and } g(\lambda_r)g(\gamma_r) < 0$$

and we also have $G(\lambda_r)$. Compute

$$d_r^P = \frac{f(\lambda_r) - (f(\gamma_r) + g(\gamma_r)(\lambda_r - \gamma_r))}{g(\gamma_r) - g(\lambda_r)}$$
$$d_r^Q = \frac{-g(\lambda_r)}{G(\lambda_r)}$$

where $|d_r^Q| = +\infty$ if $G(\lambda_r) = 0$. Now determine

$$\begin{aligned} d_r &= (\text{sign of } (-g(\lambda_r)))(\min.\{|d_r^P|, |d_r^Q|\}) \\ \nu_r &= \lambda_r + d_r. \end{aligned}$$

Terminate with the conclusion that ν_r is the minimizer of $f(\lambda)$ if either $d_r = 0$ or $g(\nu_r) = 0$.

Otherwise, update the quantities for the next step as given below. If $f(\nu_r) \geqq f(\lambda_r)$, then set $\lambda_{r+1} = \lambda_r$, $\gamma_{r+1} = \nu_r$, $G(\lambda_{r+1}) = G(\lambda_r)$. In this case there is no move in the λ_r-sequence.

If $f(\nu_r) < f(\lambda_r)$, then set $\lambda_{r+1} = \nu_r$, and

$$\begin{aligned} &\text{if } g(\lambda_r)g(\nu_r) > 0, \text{ then set } \gamma_{r+1} = \gamma_r \\ &\qquad\qquad G(\lambda_{r+1}) = \frac{g(\nu_r) - g(\lambda_r)}{d_r} \\ &\text{if } g(\lambda_r)g(\nu_r) < 0, \text{ then set } \gamma_{r+1} = \lambda_r \\ &\qquad\qquad G(\lambda_{r+1}) = \frac{g(\nu_r) - g(\gamma_r)}{\nu_r - \gamma_r}. \end{aligned}$$

Under rather general assumptions, it has been proved in [10.23] that if this algorithm does not terminate in a finite number of steps, then $f(\lambda_r) \to f(\lambda_*)$ as $r \to \infty$; and that the sequence $\{\lambda_r\}$ itself converges superlinearly to λ_*, a minimizer of $f(\lambda)$.

2 Constrained Line Minimization With Convex Functions

Need For a Constraint in Line Minimization

Let $\theta(x)$ be a real valued function defined on $\mathbf{R}^n$. In algorithms for the unconstrained minimization of $\theta(x)$, we start at a point $\overline{x} \in \mathbf{R}^n$, develop a search direction $y \in \mathbf{R}^n$, $y \neq 0$, which is a descent direction at $\overline{x}$; and then have to solve the line minimization problem of minimizing $f(\lambda) = \theta(\overline{x} + \lambda y)$ over $\lambda \geqq 0$. It has been shown that such algorithms will have desirable convergence properties if the step length $\lambda \geqq 0$ is chosen so as to satisfy

$$f(\lambda) - f(0) \leqq \omega\lambda$$

where ω is a negative number that is a positive fraction of an estimate of the directional derivative of $\theta(x)$ at $\overline{x}$ in the direction y. To satisfy this condition, we define $c(\lambda) = f(\lambda) - f(0) - \omega\lambda$, and solve the constrained line minimization problem

$$\begin{aligned} &\text{minimize} && f(\lambda) \\ &\text{subject to} && c(\lambda) \leqq 0. \end{aligned}$$

For another application of constrained line minimization, consider the general NLP

$$\begin{array}{ll} \text{minimize} & \theta(x) \\ \text{subject to} & h_i(x) \leqq 0, \ i = 1 \text{ to } m. \end{array}$$

Algorithms for solving these problems usually begin with an initial feasible point $\overline{x}$, find a descent search direction y at $\overline{x}$, and do a line minimization in that direction. Define $c(\lambda) = \max.\{h_i(\overline{x} + \lambda y) : i = 1 \text{ to } m\}$. The problem of finding the best feasible point in this search direction, leads to the constrained line minimization problem: minimize $f(\lambda)$, subject to $c(\lambda) \leqq 0$.

The Constrained Line Minimization Problem

Here we consider the constrained line minimization problem

$$\begin{array}{ll} \text{minimize} & f(\lambda) \\ \text{subject to} & c(\lambda) \leqq 0 \end{array}$$

where both functions $f(\lambda)$ and $c(\lambda)$ are convex. Let

$$\mathbf{S} = \{\lambda : c(\lambda) \leqq 0\}.$$

Since $c(\lambda)$ is convex, $\mathbf{S}$ is an interval, but it may be hard to determine $\mathbf{S}$ explicitly if $c(\lambda)$ is nonlinear. However, we assume that $\mathbf{S}$ has a nonempty interior and that a feasible point (i. e., $\lambda \in \mathbf{S}$) may be found, for example, by finding an unconstrained minimum of $c(\lambda)$.

Here we discuss a modification of the algorithm of Subsection 1 due to R. Mifflin [10.28] for solving this constrained problem.

The method generates two sequences $\{\lambda_r\}$, $\{\gamma_r\}$, where for each r, λ_r is feasible and λ_r, γ_r are on opposite sides of any constrained minimization point λ_*; γ_r is either infeasible (i. e., $c(\gamma_r) > 0$) or $f(\gamma_r) \geqq f(\lambda_r)$. The sequence $\{f(\lambda_r)\}$ is non-increasing with r. In this subsection we define

$$\begin{array}{ll} g(\lambda) \in \partial f(\lambda) & \text{if } \ c(\lambda) \leqq 0 \text{ (i. e., } \lambda \in \mathbf{S}) \\ g(\lambda) \in \partial\, c(\lambda) & \text{if } \ c(\lambda) > 0 \text{ (i. e., } \lambda \notin \mathbf{S}). \end{array}$$

We therefore have

$$\begin{array}{l} c(\lambda) \geqq c(\gamma) + g(\gamma)\,(\lambda - \gamma), \ \text{for all } \lambda, \ \text{and } \gamma \notin \mathbf{S} \\ f(\lambda) \geqq f(\gamma) + g(\gamma)(\lambda - \gamma), \ \text{for all } \lambda, \ \text{and } \gamma \in \mathbf{S}. \end{array}$$

For $\overline{\lambda}$ feasible, as in Subsection 1, we define $G(\overline{\lambda}) = (g(\overline{\lambda}) - g(t))/(\overline{\lambda} - t)$ where t is feasible and $\overline{\lambda}$ is between t and λ_*. The quadratic approximation step at a feasible point $\overline{\lambda}$ is defined as before, using $G(\overline{\lambda})$.

In Step r of the algorithm, if both the points λ_r and γ_r are feasible, the polyhedral approximation step is defined exactly as under Subsection 1.

Given λ_r, γ_r, if γ_r is infeasible; then $g(\gamma_r)$ is a subgradient of the constraint function $c(\lambda)$, and is not related to the objective function $f(\lambda)$. Thus, in this case, the polyhedral approximation step given λ_r, γ_r is not well defined as in Subsection 1. One aim for this step could be to move the γ-sequence towards feasibility. Taking this step to be $\hat{d}$, where $\lambda_r + \hat{d} = \gamma_r - (c(\gamma_r)/g(\gamma_r))$ would correspond to a Newton-Raphson step for solving $c(\lambda) = 0$ based upon linearization of $c(\lambda)$ at γ_r. On the other hand, in order to make a move not just towards feasibility, but towards a minimizing feasible point, we could take the step to be $\tilde{d}$ where $\lambda_r + \tilde{d}$ is the point at which the linearization of $f(\lambda)$ at λ_r, and the linearization of $c(\lambda)$ at γ_r become equal. This leads to $\tilde{d} = (-c(\gamma_r) - g(\gamma_r)(\lambda_r - \gamma_r))/(g(\gamma_r) - g(\lambda_r))$. In order to achieve fast convergence, the actual polyhedral approximation step in this case, from the feasible point λ_r, is taken to be a compromise between $\hat{d}$ and $\tilde{d}$ given by

$$d_r^P = \frac{P(\lambda_r, \gamma_r)}{g(\gamma_r) - b_r g(\lambda_r)}$$

where $P(\lambda_r, \gamma_r) = -c(\gamma_r) - g(\gamma_r)(\lambda_r - \gamma_r)$ and $b_r = P(\lambda_r, \gamma_r)$. We are now ready to describe the algorithm.

The algorithm needs an initial pair of points λ_1, γ_1 such that λ_1 is feasible (i. e., $\lambda_1 \in \mathbf{S}$), and either $c(\gamma_1) > 0$ or $f(\gamma_1) \geqq f(\lambda_1)$; and $g(\lambda_1)g(\gamma_1) < 0$. This implies that a constrained minimizing point lies between λ_1 and γ_1. Also choose $G(\lambda_1) \geqq 0$. We will now describe the general step in the algorithm.

Step r. Let λ_r, γ_r be the points at the beginning of this step. Define

$$P(\lambda_r, \gamma_r) = -c(\gamma_r) - g(\gamma_r)(\lambda_r - \gamma_r), \text{ and } b_r = P(\lambda_r, \gamma_r), \text{ if } c(\gamma_r) > 0$$
$$P(\lambda_r, \gamma_r) = f(\lambda_r) - f(\gamma_r) - g(\gamma_r)(\lambda_r - \gamma_r), \text{ and } b_r = 1, \text{ if } c(\gamma_r) \leqq 0$$
$$d_r^P = \frac{P(\lambda_r, \gamma_r)}{g(\gamma_r) - b_r g(\lambda_r)}$$
$$d_r^Q = \frac{-g(\lambda_r)}{G(\lambda_r)}, \text{ if } G(\lambda_r) > 0$$
$$|d_r^Q| = +\infty, \text{ if } G(\lambda_r) \leqq 0$$
$$d_r = (\text{sign of } (-g(\lambda_r)))(\min.\{|d_r^P|, |d_r^Q|\})$$
$$\nu_r = \lambda_r + d_r.$$

Terminate with the conclusion that ν_r is the optimum solution of the problem if either $d_r = 0$ or $g(\nu_r) = 0$.

Otherwise, update the quantities for the next step as given below. If $c(\nu_r) > 0$ or $f(\nu_r) \geqq f(\lambda_r)$, set $\lambda_{r+1} = \lambda_r$, $\gamma_{r+1} = \nu_r$, $G(\lambda_{r+1}) = G(\lambda_r)$.

If $c(\nu_r) \leqq 0$ and $f(\nu_r) < f(\lambda_r)$, then set $\lambda_{r+1} = \nu_r$, and

$$\text{if } g(\lambda_r)g(\nu_r) > 0, \text{ then set } \gamma_{r+1} = \gamma_r, G(\lambda_{r+1}) = \frac{g(\nu_r) - g(\lambda_r)}{d_r}$$
$$\text{if } g(\lambda_r)g(\nu_r) < 0, \text{ then set } \gamma_{r+1} = \lambda_r, G(\lambda_{r+1}) = \frac{g(\nu_r) - g(\gamma_r)}{\nu_r - \gamma_r}.$$

Under rather general conditions, R. Mifflin [10.28] has proved that if the algorithm does not terminate finitely, then $f(\lambda_r)$ converges to the minimum value of $f(\lambda)$ over $\mathbf{S}$, and that the sequence $\{\lambda_r\}$ itself converges to an optimum solution of the problem, λ_*, with $|\lambda_r - \lambda_*||\gamma_r - \lambda_r|$ converging to zero superlinarly.

3 General Constrained Line Minimization

Let $f(\lambda)$, $c(\lambda)$ be real valued functions defined on $\mathbf{R}^1$, not necessarily convex. Here we consider the constrained line minimization problem

$$\begin{array}{ll} \text{minimize} & f(\lambda) \\ \text{subject to} & c(\lambda) \leqq 0. \end{array}$$

The set $\mathbf{S} = \{\lambda : c(\lambda) \leqq 0\}$ is the feasible set. Since $c(\lambda)$ is not assumed to be convex, $\mathbf{S}$ may consist of a collection of disjoint intervals.

Let $F(\lambda)$ denote either $f(\lambda)$ or $c(\lambda)$. If $F(\lambda)$ is continuoulsy differentiable at λ, we let $\partial F(\lambda)$ be the singleton set $\{\frac{dF(\lambda)}{d\lambda}\}$, as in Appendix 3. If $F(\lambda)$ is not differentiable at λ, $\partial F(\lambda)$ denotes the set of subgradients or generalized gradients, it is the convex hull of all limits of sequences of the form $\{\frac{dF(\lambda_k)}{d\lambda} : \{\lambda_k\} \to \lambda$ and $F(\lambda)$ is differentiable at each $\lambda_k\}$. With this definition $\partial F(\lambda)$ agrees with the subdifferential set when $F(\lambda)$ is convex. Also if $F(\lambda)$ is not given explicitly, but is defined implicitly as the pointwise supremum, say, as $F(\lambda) = \max\{F_1(\lambda), \dots, F_t(\lambda)\}$ where each $F_i(\lambda)$ is continuously differentiable, then $\partial F(\lambda)$ will be the convex hull of $\{\frac{dF_i(\lambda)}{d\lambda}$: over all i such that $F_i(\lambda) = F(\lambda)\}$. The algorithm discussed in this subsection needs a subroutine which can evaluate $F(\lambda)$ for any λ, and another subroutine to obtain a number $g(\lambda) \in \partial F(\lambda)$.

Stationary Points

A point $\lambda_* \in \mathbf{S}$ is a stationary point for this constrained line minimization problem if

$$\begin{aligned} \text{either } c(\lambda_*) < 0, &\text{ and } 0 \in \partial f(\lambda_*) \\ \text{or } c(\lambda_*) = 0, &\text{ and } 0 \in \text{ convex hull of } \partial f(\lambda_*) \cup \partial c(\lambda_*) \end{aligned}$$

because these are the necessary optimality conditions for this problem. See Figure 10.7.

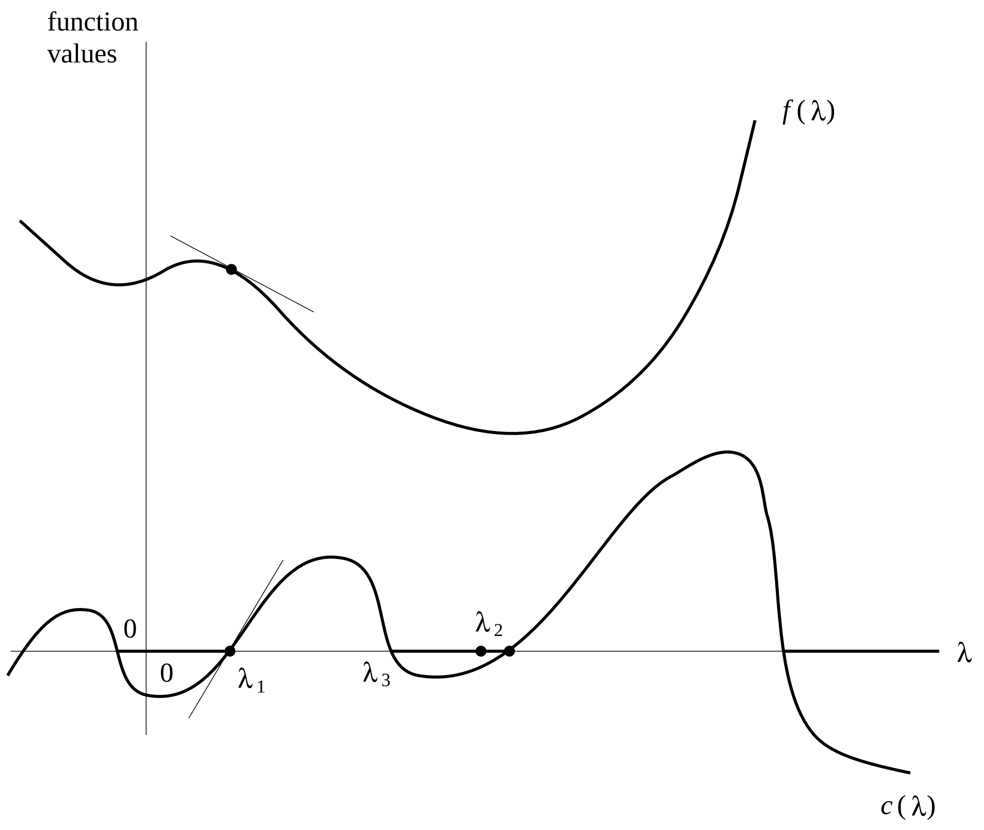

Figure 10.7 The feasible set **S** consists of the thick portion of the λ-axis. λ_1 is a stationary point since $c(\lambda_1) = 0$; $\frac{dc(\lambda_1)}{d\lambda_1} > 0$, $\frac{df(\lambda_1)}{d\lambda} < 0$ and so 0 is in the convex hull of $\frac{dc(\lambda_1)}{d\lambda}$, and $\frac{df(\lambda_1)}{d\lambda}$. λ_2 is another stationary point, but λ_3 is not.

In the algorithm discussed in this subsection, we need

$$g(\lambda) \in \partial f(\lambda) \text{ if } \lambda \in \mathbf{S}$$
$$g(\lambda) \in \partial c(\lambda) \text{ if } \lambda \notin \mathbf{S}.$$

The algorithm generates two sequences of points $\{\lambda_r\}$, $\{\gamma_r\}$ with λ_r feasible for all r and $f(\lambda_r)$ non-increasing. For each r we will have

$$c(\lambda_r) \leqq 0 \text{ and } g(\lambda_r)(\gamma_r - \lambda_r) < 0$$
$$\text{and either } c(\gamma_r) > 0, \text{ or } c(\gamma_r) \leqq 0 \text{ and } f(\gamma_r) \geqq f(\lambda_r).$$

These conditions imply that there exists a stationary point between λ_r and γ_r. The algorithm needs a pair of initial points λ_1, γ_1 satisfying the above conditions, these can be obtained by a suitable initialization routine. The sequence of points $\{\lambda_r\}$ obtained in the algorithm converges to a stationary point λ_* and $|\lambda_r - \lambda_*|$. $|\gamma_r - \lambda_*|$ converges to zero superlinearly.

The Quadratic Approximation Step

As before, $G(\lambda_r)$ is an approximation to the second derivative of $f(\lambda)$ at λ_r, and it is determined in a one-sided secant-manner, that is, when $\lambda_r \neq \lambda_1$,

$$G(\lambda_r) = \frac{g(\lambda_r) - g(t_r)}{\lambda_r - t_r}$$

where t_r is a feasible λ_j or γ_j for some $j < r$ and is on the opposite side of λ_r from γ_r. If $f(\lambda)$ is convex, then we will have $G(\lambda_r) \geqq 0$ for all r. But due to nonconvexity we may get some $G(\lambda_r) \leqq 0$. So, the quadratic approximation step is defined here by

$$d_r^Q = \frac{-g(\lambda_r)}{\max.\{G(\lambda_r), 0\}}$$

with the understanding that $|d_r^Q| = +\infty$ if $G(\lambda_r) \leqq 0$.

The Polyhedral Approximation Step

Consider the case when both λ_r and γ_r are feasible first. In this case, if $g(\lambda_r)$, $g(\gamma_r)$ have opposite signs, we define the polyhedral approximation step by

$$d_r^P = \frac{P(\lambda_r, \gamma_r)}{g(\gamma_r) - g(\lambda_r)}$$

where $P(\lambda_r, \gamma_r) = f(\lambda_r) - f(\gamma_r) - g(\gamma_r)(\lambda_r - \gamma_r)$, as before. If $P(\lambda_r, \gamma_r) \geqq 0$ (which will be the case when $f(\lambda)$ is convex) then $\lambda_r + d_r^P$ will be between λ_r and γ_r. Due to nonconvexity it may happen that $g(\lambda_r)$ and $g(\gamma_r)$ do not have opposite signs and/or $P(\lambda_r, \gamma_r)$ is negative. In this case, the polyhedral approximation step needs to be modified as follows. See Figure 10.8.

Let H_r be a secant estimate of $f''(\lambda)$ near γ_r, that is when $\gamma_r \neq \gamma_1$,

$$H_r = \frac{g(\gamma_r) - g(u_r)}{\gamma_r - u_r}$$

where u_r is a feasible λ_j or γ_j for some $j < r$ on the opposite side of γ_r from λ_r. In this case a quadratic approximation to $f(\lambda)$ around γ_r is

$$q(\lambda) = f(\gamma_r) + g(\gamma_r)(\lambda - \gamma_r) + \frac{1}{2}H_r(\lambda - \gamma_r)^2.$$

A linear approximation for $q(\lambda_r + d)$ based at λ_r is

$$f(\gamma_r) + g(\gamma_r)(\lambda_r - \gamma_r) + \frac{1}{2}H_r(\lambda_r - \gamma_r)^2 + [g(\gamma_r) + H_r(\lambda_r - \gamma_r)]d.$$

We can take d_r^P to be the value of d which equalizes this to $f(\lambda_r)+dg(\lambda_r)$. This leads to $d = (f(\lambda_r) - f(\gamma_r) - [g(\gamma_r)+h](\lambda_r - \gamma_r))/(g(\gamma_r)+2h-g(\lambda_r))$, where $h = \frac{1}{2}H_r(\lambda_r - \gamma_r)$. Since this needs to be carried out only under negative curvature, we define a negative curvature correction

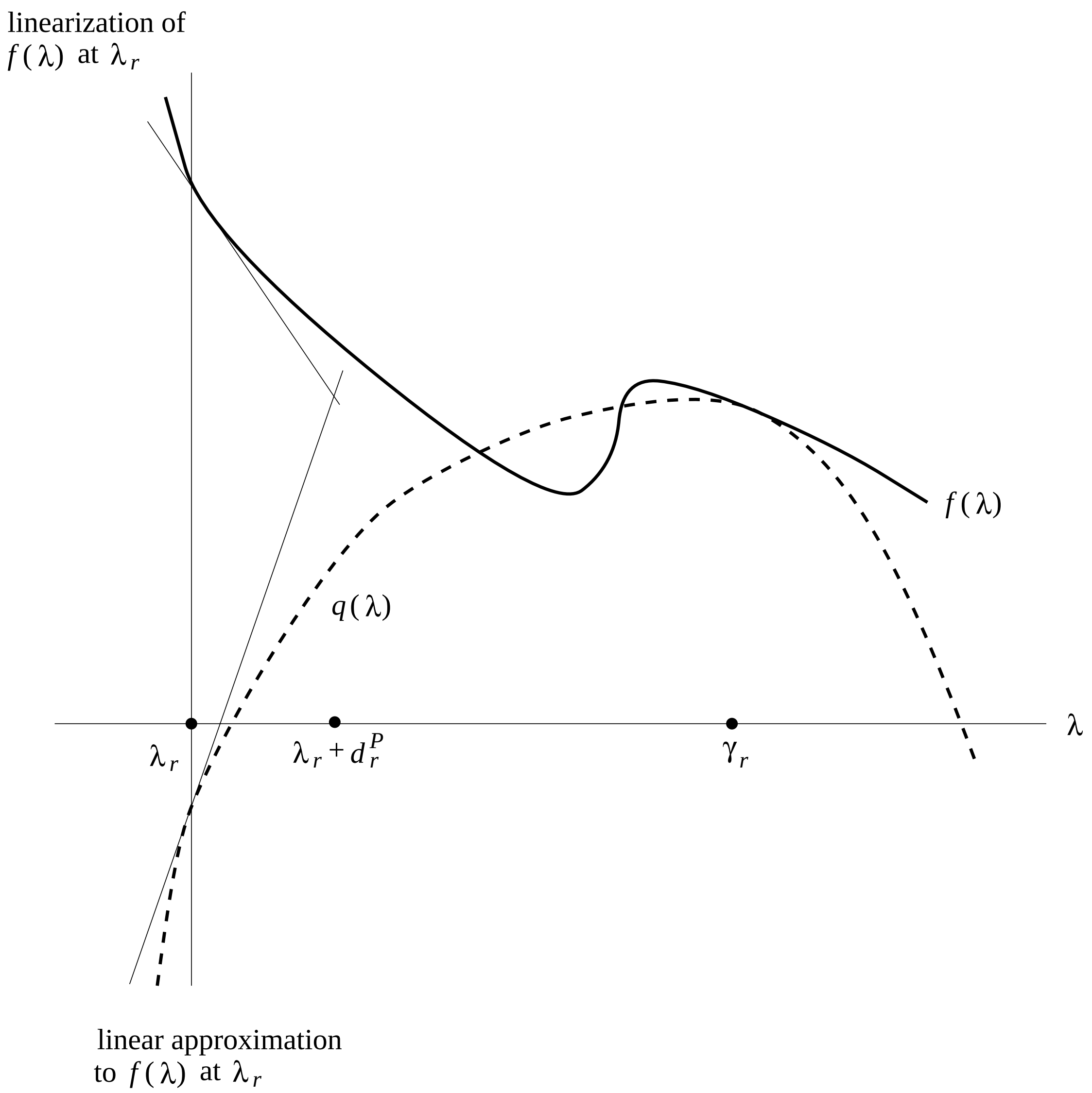
linearization of $f(\lambda)$ at λ_r
$f(\lambda)$
$q(\lambda)$
λ
λ_r
$\lambda_r + d_r^P$
γ_r
linear approximation to $f(\lambda)$ at λ_r

Figure 10.8 $q(\lambda)$ (the dashed curve) is the quadratic approximation to $f(\lambda)$ based at $\lambda = \gamma_r$. The point $\lambda_r + d_r^P$ is the point where the linearizations of $f(\lambda)$ and $q(\lambda)$ based at λ_r, become equal.

$$h_r = \frac{1}{2}(\lambda_r - \gamma_r)\min\{H_r, 0\}$$

and let

$$P_r = f(\lambda_r) - f(\gamma_r) - (g(\gamma_r) + h_r)(\lambda_r - \gamma_r)$$

$$d_r^P \begin{cases} = 0, & \text{if } P_r \leqq 0 \\ = P_r/(g(\gamma_r) + 2h_r - g(\lambda_r)), & \text{if } P_r > 0. \end{cases}$$

Now consider the case when $\gamma_r \notin \mathbf{S}$. In this case we make a similar quadratic approximation to $c(\lambda)$, and using it estimate the point $\lambda_r + d$ where $c(\lambda_r + d)$ would be zero. In this case H_r is an estimate of $c''(\gamma_r)$, and h_r is defined as above. Using again the compromise as done in Subsection 2 for fast convergence, in this case we are lead to the following polyhedral approximation step.

$$P_r = -c(\gamma_r) - (g(\gamma_r) + h_r)(\lambda_r - \gamma_r)$$

$$d_r^P \begin{cases} = 0, & \text{if } P_r \leqq 0 \\ = P_r/(g(\gamma_r) + 2h_r - P_r g(\lambda_r)), & \text{if } P_r > 0. \end{cases}$$

To handle this general problem, we also define a positive safeguard parameter ζ such that

$$\zeta|\gamma_r - \lambda_r| < \frac{1}{2}$$

so that

$$\zeta(\gamma_r - \lambda_r)^2 < \frac{1}{2}|\gamma_r - \lambda_r| < |\gamma_r - \lambda_r| - \zeta|\gamma_r - \lambda_r|^2.$$

In the algorithm, the step d_r^P is modified into d_r^ζ so that $|d_r^\zeta|$ is between the lower and upper bounds in the above inequality. This guarantees that $\lambda_r + d_r$ is away from λ_r and γ_r. If the problem functions are convex, then $G(\lambda_r) \geqq 0$, $H_r \geqq 0$, $h_r = 0$ and $P_r \geqq 0$, and if $\zeta = 0$, the algorithm discussed below will be the same as the one discussed in Subsection 2. Now we describe the algorithm.

The algorithm needs an initial pair of points λ_1 and γ_1 satisfying the conditions mentioned above. Choose the safeguard parameter $\zeta > 0$ such that $\zeta(\gamma_1 - \lambda_1) < \frac{1}{2}$, and choose the initial curvature estimates $G(\lambda_1)$ and H_1. We will now describe the general step in the algorithm.

Step r. Let λ_r, γ_r be the points at the beginning of this step. Let $G(\lambda_r)$, H_r be the curvature estimates. Set

$$h_r = \frac{1}{2}(\lambda_r - \gamma_r)\min\{H_r, 0\}.$$

$$\left.\begin{aligned} P_r &= -c(\gamma_r) - (g(\gamma_r) + h_r)(\lambda_r - \gamma_r) \\ \text{and } \delta_r &= g(\gamma_r) + 2h_r - P_r g(\lambda_r) \end{aligned}\right\} \quad \text{if } c(\gamma_r) > 0.$$

$$\left.\begin{aligned} P_r &= f(\lambda_r) - f(\gamma_r) - (g(\gamma_r) + h_r)(\lambda_r - \gamma_r) \\ \delta_r &= g(\gamma_r) + 2h_r - g(\lambda_r) \end{aligned}\right\} \quad \text{if } c(\gamma_r) \leqq 0.$$

$$\begin{aligned} d_r^P &= 0 && \text{if } P_r \leqq 0 \\ &= P_r/\delta_r && \text{if } P_r > 0 \\ \zeta_r &= \zeta|\gamma_r - \lambda_r|^2 \\ \alpha_r &= +1 && \text{if } -g(\lambda_r) > 0 \\ &= -1 && \text{if } -g(\lambda_r) < 0. \end{aligned}$$

$$\begin{aligned} d_r^\zeta &= \alpha_r \zeta_r, && \text{if } |d_r^P| < \zeta_r \\ &= \alpha_r |d_r^P|, && \text{if } \zeta_r \leqq |d_r^P| \leqq |\gamma_r - \lambda_r| - \zeta_r \\ &= \alpha_r(|\gamma_r - \lambda_r| - \zeta_r), && \text{if } |d_r^P| > |\gamma_r - \lambda_r| - \zeta_r \end{aligned}$$

$$\begin{aligned} d_r^Q &= -g(\lambda_r)/\max\{G(\lambda_r), 0\} \\ d_r &= \alpha_r(\min\{|d_r^Q|, |d^\zeta|\}) \\ \nu_r &= \lambda_r + d_r. \end{aligned}$$

If $c(\nu_r) \leqq 0$, $f(\nu_r) < f(\lambda_r)$ and $g(\nu_r) = 0$, terminate with the conclusion that ν_r is a stationary point.

Otherwise update the quantities for the next step as given below.

If $c(\nu_r) > 0$, or $f(\nu_r) \geqq f(\lambda_r)$, then set $\lambda_{r+1} = \lambda_r$, $\gamma_{r+1} = \nu_r$, $G(\lambda_{r+1}) = G(\lambda_r)$, and $H_{r+1} = [g(\nu_r) - g(\gamma_r)]/(\nu_r - \gamma_r)$.

If $c(\nu_r) \leqq 0$, $f(\nu_r) < f(\lambda_r)$, $g(\nu_r) \neq 0$, and $g(\lambda_r)g(\nu_r) > 0$, then set $\lambda_{r+1} = \nu_r$, $\gamma_{r+1} = \gamma_r$, $G(\lambda_{r+1}) = (g(\nu_r) - g(\lambda_r))/d_r$, and $H_{r+1} = H_r$.

If $c(\nu_r) \leqq 0$, $f(\nu_r) < f(\lambda_r)$, $g(\nu_r) \neq 0$, and $g(\lambda_r)g(\nu_r) < 0$, then set $\lambda_{r+1} = \nu_r$, $\gamma_{r+1} = \lambda_r$, $G(\lambda_{r+1}) = [g(\nu_r) - g(\gamma_r)]/(\nu_r - \gamma_r)$, and $H_{r+1} = G(\lambda_r)$.

Under rather general conditions on the functions, R. Mifflin [10.29] has proved that if the algorithm does not terminate finitely, then $\{\lambda_r\}$ converges to a stationary point of $f(\lambda)$ on S.

To start the algorithm from a feasible λ_1 when a suitable γ_1 is not known, one can use a safeguarded quadratic step of the form

$$-g(\lambda_j)/\max[G(\lambda_j), a_j],\ j = 1, 2$$

where $\{a_j\}$ is a bounded positive sequence chosen so that it converges to zero if $\{g(\lambda_j)\}$ converges to zero.

10.8 SURVEY OF DESCENT METHODS FOR UNCONSTRAINED MINIMIZATION IN $\mathbf{R}^n$

In this section we consider methods for solving the problem

$$\begin{array}{ll} \text{minimize} & \theta(x) \\ \text{over} & x \in \mathbf{R}^n \end{array} \tag{10.4}$$

where $\theta(x)$ is a real valued continuously differentiable function defined over $\mathbf{R}^n$. The methods discussed in this section make use of the first and sometimes the second order partial derivatives of $\theta(x)$ when they exist, or approximations for these constructed from the information accumulated over the iterations. The methods are iterative, they generate a sequence of points $\{x^0, x^1, x^2, \ldots\} \subset \mathbf{R}^n$ beginning with an initial point x^0, and satisfy the property that $\theta(x^r)$ monotonically decreases as r increases.

In this section $\nabla\theta(x^r)$ denotes the row vector of partial derivatives of $\theta(x)$ at the point x^r (the gradient vector of $\theta(x)$ at x^r). When the second order partial derivatives exist, we denote the $n \times n$ Hessian matrix of $\theta(x)$ at the point x^r by $H(\theta(x^r)) =$

$(\frac{\partial^2\theta(x^r)}{\partial x_i \partial x_j})$. In the methods discussed in this section, each iteration or step consists of three parts. The $(k+1)$th step begins with the point x^k (x^k is the point obtained at the end of step k if $k > 0$, x^0 is the initial point with which the method is initiated) and consists of the following parts

i) compute the search direction at x^k, denoted by y^k. $y^k \in \mathbf{R}^n$, $y^k \neq 0$,
ii) compute the step length in the search direction, $\alpha_k > 0$,
iii) compute the new point $x^{k+1} = x^k + \alpha_k y^k$ and check whether termination criteria are satisfied. If the termination criteria are satisfied, x^{k+1} is accepted as the solution of (10.4). Otherwise, continue the method by going to the next step.

In order to guarantee that $\theta(x^r)$ decreases monotonically, we require the search directions to be **descent directions**. The point $y \in \mathbf{R}^n$, $y \neq 0$ is said to be a descent direction for $\theta(x)$ at the point x^k if there exists a $\overline{\lambda} > 0$ for which

$$\theta(x^k + \lambda y) < \theta(x^k), \quad \text{for all } 0 < \lambda \leqq \overline{\lambda}. \tag{10.5}$$

Since $\theta(x)$ is differentiable at x^k, (10.5) implies that the limit of $(\theta(x^k+\lambda y) - \theta(x^k))/\lambda$ as λ approaches zero through positive values, is $\leqq 0$, that is $(\nabla\theta(x^k))y \leqq 0$. Conversely, it can be verified that any y satisfying

$$(\nabla\theta(x^k))y < 0 \tag{10.6}$$

is a descent direction at x^k. The condition (10.6) is a sufficient condition for y to be a descent direction at x^k. We define a descent direction for $\theta(x)$ at x^k, to be a $y \in \mathbf{R}^n$, $y \neq 0$, satisfying (10.6). Similarly the point $y \in \mathbf{R}^n$, $y \neq 0$ is said to be a **nonascent direction** for $\theta(x)$ at x^k if,

$$(\nabla\theta(x^k))^T y \leqq 0. \tag{10.7}$$

When $\theta(x)$ is twice continuously differentiable, the point $y \in \mathbf{R}^n$, $y \neq 0$ is said to be a **direction of nonpositive curvature** for $\theta(x)$ at x^k if,

$$y^T H(\theta(x^k))y \leqq 0 \tag{10.8}$$

and a **direction of negative curvature** if,

$$y^T H(\theta(x^k))y < 0. \tag{10.9}$$

10.8.1 How to Determine the Step Length?

Let x^k be the current point and suppose the search direction y^k, which is a descent direction, has been selected. Since x^k, y^k are given points, $\theta(x^k+\lambda y^k)$ is now a function of λ only, and it can be verified that its derivative with respect to λ is $(\nabla\theta(x^k+\lambda y^k))y^k$. The descent step length can be determined to minimize $\theta(x^k + \lambda y^k)$ over $\lambda \geqq 0$. This operation is a line search operation. Step lengths determined to minimize $\theta(x^k + \lambda y^k)$

over $\lambda \geqq 0$ are referred to as **optimal step lengths** and algorithms using them are called **optimal step descent techniques**. Since y^k is a descent direction for $\theta(x)$ at x^k, the optimal step length λ_k is > 0 and

$$\frac{d\theta}{d\lambda}(x^k + \lambda_k y^k) = (\nabla\theta(x^k + \lambda_k y^k))y^k = 0. \tag{10.10}$$

So if optimal step lengths are used, the gradient direction at the termination of a line search step is orthogonal to the descent direction.

In practice, it may not be efficient to use optimal step lengths in every iteration. Algorithms which allow for termination of line searches when conditions for an approximate minimum on the line are satisfied, are said to use **partial** or **inexact line searches**. When using inexact line searches, it is necessary to make sure that the line search achieves a sufficient decrease in objective value, to guarantee convergence. A practical criterion requires that the step length λ be determined to make $|(\nabla\theta(x^k + \lambda y^k))y^k|$ sufficiently small. Stated in terms of the decrease in the magnitude of the derivative of $\theta(x^k + \lambda y^k)$ with respect to λ from that at $\lambda = 0$, another criterion requires that the step length λ be chosen to satisfy

$$|(\nabla\theta(x^k + \lambda y^k))y^k| \leqq \eta|(\nabla\theta(x^k))y^k| \tag{10.11}$$

where η is a parameter satisfying $0 \leqq \eta < 1$. If $\eta = 0$ in (10.11), exact line searches are required, and when η is small, the line search procedure needs to be close to optimal.

Step Length Criterion to Achieve Sufficient Rate of Decrease

A fundamental requirement of step size procedures used in descent methods is that there be a sufficient decrease in the objective value in each step. There are many ways of specifying what a "sufficient-decrease" is. For example, consider the line minimization problem of minimizing $\theta(x^k + \lambda y^k)$ over $\lambda \geqq 0$, where y^k is a descent direction for $\theta(x)$ at x^k. The quantity, $(\nabla\theta(x^k))y^k$, the directional derivative of $\theta(x)$ in the search direction y^k, is a measure of the rate of decrease in $\theta(x)$ at x^k in the direction y^k. Select a number α, $0 < \alpha < 1$, known as the **attenuation factor**. One sufficient decrease criterion requires that over the step length taken, the function value must decrease per unit step, at least a fraction α of the rate of decrease in $\theta(x)$ at x^k in the direction y^k. That is, that the step length λ chosen satisfy

$$\theta(x^k) - \theta(x^k + \lambda y^k) \geqq \lambda\alpha|(\nabla\theta(x^k))y^k|. \tag{10.12}$$

To depict this pictorially, we plot λ on the horizontal axis, and function values on the vertical axis in the two dimensional cartesian plane in Figure 10.9. The curve in Figure 10.9 is $\theta(x^k + \lambda y^k)$ plotted against λ. The straight lines in Figure 10.9 are plots of

$$l_\alpha(\lambda) = \theta(x^k) - \lambda\alpha|(\nabla\theta(x^k))y^k|$$

against λ.

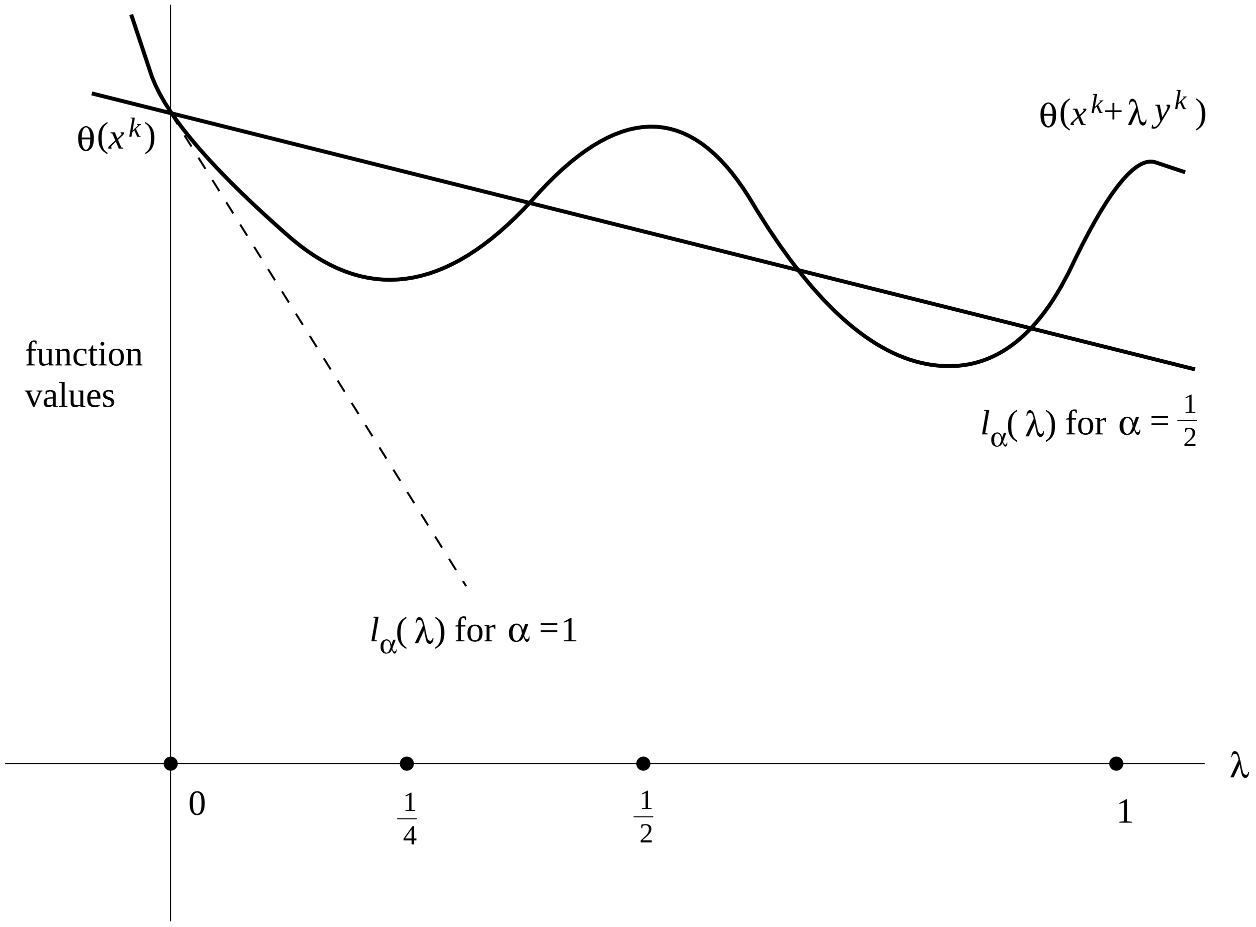
$\theta(x^k)$
$\theta(x^k+\lambda y^k)$
function values
$l_\alpha(\lambda)$ for $\alpha = \frac{1}{2}$
$l_\alpha(\lambda)$ for $\alpha = 1$
0
$\frac{1}{4}$
$\frac{1}{2}$
1
λ

Figure 10.9 The dashed line is $l_\alpha(\lambda)$ for $\alpha = 1$. The continuous straight line is $l_\alpha(\lambda)$ for $\alpha = \frac{1}{2}$. (10.12) requires that for step length λ chosen, $\theta(x^k + \lambda y^k) \leqq l_\alpha(\lambda)$.

The sufficient decrease condition (10.12) states that the step size λ chosen, should satisfy

$$\theta(x^k + \lambda y^k) \leqq l_\alpha(\lambda) = \theta(x^k) - \lambda\alpha|(\nabla\theta(x^k))y^k|.$$

This inequality is called Armijo inequality.

Other Step Length Criteria

Many theoretical convergence proofs for descent algorithms assume that the step length used is the first local minimum along the line in the direction $\lambda \geqq 0$.

The First Order Armijo Step Size Procedure

This procedure was introduced by L. Armijo [10.1]. Let y^k be the descent direction for $\theta(x)$ at the current point x^k. Let $0 < \alpha < 1$ be a predetermined constant. This procedure finds $s =$ smallest nonnegative integer satisfying

$$\theta(x^k) - \theta\big(x^k + \frac{y^k}{2^{\mathrm{s}}}\big) \geqq -\frac{\alpha}{2^{\mathrm{s}}}(\nabla\theta(x^k))y^k \tag{10.13}$$

and chooses the step length to be $1/2^{\mathrm{s}}$. Since y^k is a descent direction, a finite s satisfying (10.13) exists.

As an example, consider the problem depicted in Figure 10.9. Let the attenuation factor $\alpha = \frac{1}{2}$. In Figure 10.9, we verify that $\lambda = 1$ violates the Armijo inequality (10.13), since $\theta(x^k + y^k) > l_\alpha(1)$ for $\alpha = \frac{1}{2}$. Even $\lambda = \frac{1}{2}$ violates the Armijo inequality (10.13) since $\theta(x^k + \frac{1}{2}y^k) > l_\alpha(\frac{1}{2})$ for $\alpha = \frac{1}{2}$. $\lambda = \frac{1}{4}$ satisfies Armijo inequality (10.13) because $\theta(x^k + \frac{1}{4}y^k) < l_\alpha(\frac{1}{4})$ for $\alpha = \frac{1}{2}$ in Figure 10.9. So the step length chosen by this procedure in this problem is $\lambda_k = \frac{1}{4}$.

It can be verified that there is always a positive integer s satisfying (10.13). So, the step length indicated in this procedure is well defined and unique.

One thing that should be noted here is that the step length chosen by this procedure depends on the scaling of y^k. Replacing y^k by βy^k where $\beta > 0$; does not change the search direction, or the line search problem; but it could change the step length chosen by this procedure and the final point obtained in the line search by this procedure. The direction y^k is usually selected by a descent direction selection subroutine, using the values of the function $\theta(x)$ or its gradient vectors or hessian matrices evaluated at previous points, and this procedure takes the output of that subroutine as it is.

Second Order Armijo Step Size Procedure

This procedure is useful when using second order methods like Newton's method discussed below. Let x^k be the current point. Here we will have two directions of nonascent, y^k and h^k. If $\nabla\theta(x^k) \neq 0$, y^k should be a descent direction satisfying (10.6). If $\nabla\theta(x^k) = 0$, then h^k is a direction of negative curvature satisfying (10.9). Let $0 < \alpha < 1$ be a predetermined constant. Let s be the smallest nonnegative integer s satisfying

$$\theta(x^k) - \theta\Big(x^k + \frac{y^k}{2^{\mathrm{s}}} + \frac{h^k}{2^{\mathrm{s}/2}}\Big) \geqq -\frac{\alpha}{2^{\mathrm{s}}}((\nabla\theta(x^k))y^k + \frac{1}{2}(h^k)^T H(\theta(x^k))h^k) \tag{10.14}$$

and take the next point to be $x^{k+1} = x^k + \frac{y^k}{2^{\mathrm{s}}} + \frac{h^k}{2^{\mathrm{s}/2}}$. The conditions mentioned above guarantee that a finite s satisfying (10.14) exists. It can be proved that if $\theta(x)$ is twice continuously differentiable, and a descent algorithm using this second order Armijo procedure is carried out, then every limit point $\overline{x}$ of the sequences of points $\{x^r\}$ generated by this method is a point $\overline{x}$ satisfying $\nabla\theta(\overline{x}) = 0$ and $H(\theta(\overline{x}))$ is PSD (the second order necessary optimality condition for $\overline{x}$ to be a local minimum of $\theta(x)$). See [10.1, 10.26, 10.27].

10.8.2 The Various Methods

Now we present various descent methods for (10.4). Since each method has the same structure (each step consisting of three parts (i), (ii), (iii) described under Section 10.8), we will briefly describe how the search direction is chosen in each step, what step size procedures can be used, and a summary of convergence results.

10.8.3 The Method of Steepest Descent

In this method, the search direction in the $(k+1)$th step is chosen to be the steepest descent direction at the current point x^k. The steepest descent direction at x^k is clearly the direction $d \in \mathbf{R}^n$ which minimizes

$$\operatorname*{limit}_{\lambda \to 0^+} \frac{\theta(x^k + \lambda d) - \theta(x^k)}{\lambda} = (\nabla\theta(x^k))d$$

subject to $\|d\| = 1$. In $\mathbf{R}^n$, $\|d\|$, the distance between d and 0 can be measured by the general distance function $f(d) = \sqrt{d^T A d}$ where A is a PD symmetric matrix of order n. If $A = I$, $f(d)$ becomes the usual Euclidean distance. The matrix A is known as the metric matrix in the distance function $f(d)$. With respect to this metric matrix A, the steepest descent direction at x^k is therefore the d which minimizes $(\nabla\theta(x^k))d$ subject to $\|d\| = d^T A d = 1$. It can be verified that this direction is given by $-(\nabla\theta(x^k))A^{-1}$ if $\nabla\theta(x^k) \neq 0$.

The steepest descent method, dating back to Cauchy (1847) takes the metric matrix to be I in each step, and thus uses the search direction to be $y^k = -(\nabla\theta(x^k))^T$ when x^k is the current point.

It can be shown that the steepest descent method converges when applied with any of the step length procedures discussed in Section 10.8.1. Every limit point $\overline{x}$ of the sequence $\{x^r\}$ generated satisfies the necessary optimality condition $\nabla\theta(\overline{x}) = 0$. The convergence rate for the algorithm is linear [10.13, 10.17, 10.27, 10.33]. In practice, the method has been observed to be notoriously slow and unreliable due to round-off effects.

10.8.4 Newton's Method

This is a second derivative method that can be used only if $\theta(x)$ is twice continuously differentiable. At the current point x^k, $\theta(x)$ is approximated by the quadratic function $\theta(x^k) + (\nabla\theta(x^k))(x - x^k) + \frac{1}{2}(x - x^k)^T H(\theta(x^k))(x - x^k)$, containing the first three terms of the Taylor series expansion for $\theta(x)$ around x^k. The first order necessary condition for the minimum $y = x - x^k$ of this quadratic approximation is that y satisfies

$$H(\theta(x^k))y = -(\nabla\theta(x^k))^T. \tag{10.15}$$

A direction y that satisfies (10.15) is known as the **Newton direction** for $\theta(x)$ and x^k. Assuming the $H(\theta(x^k))$ is PD (since this matrix is nonsingular, the solution of (10.15) is unique), the unique minimum of the quadratic approximation at x^k is

$$x^{k+1} = x^k - (H(\theta(x^k)))^{-1}(\nabla\theta(x^k))^T \tag{10.16}$$

the iterative scheme given by (10.16) is the traditional Newton's method. When $H(\theta(x^k))$ is PD, $(H(\theta(x^k)))^{-1}(\nabla\theta(x^k))^T$ is the steepest descent direction at x^k using $H(\theta(x^k))$ as the metric matrix, and the formula (10.16) is based on using a constant step length of $+1$ in this direction. When $\theta(x)$ satisfies the property that $H(\theta(x))$ is PD for all x (in this case $\theta(x)$ is strictly convex) Newton's method uses the steepest descent direction with the metric matrix $H(\theta(x^k))$ in the step in which x^k is the current point, and since the metric matrix changes in each step, it is called a **variable metric method** in this case.

As an illustration of convergence proofs we provide below a theorem on the convergence of Newton's method.

Theorem 10.1 *Suppose $\theta(x)$ is twice continuously differentiable. Let $H(x) = (h_{ij}(x)) = H(\theta(x))$. So $h_{ij}(x) = \frac{\partial^2\theta(x)}{\partial x_i \partial x_j}$. Suppose each of the functions $h_{ij}(x)$ satisfies the Lipschitz condition, that is, there exists a positive number α satisfying $|h_{ij}(\xi) - h_{ij}(\eta)| \leqq \alpha\|\xi - \eta\|$ for all $\xi, \eta \in \mathbf{R}^n$. Let $\overline{x}$ be a point satisfying $\nabla\theta(\overline{x}) = 0$, $H(\theta(x))$ is PD. If the initial point x^0 is sufficiently close to $\overline{x}$, the sequence of points $\{x^r : r = 0, 1, \dots\}$ obtained by Newton's method converges to $\overline{x}$ at a second order rate.*

Proof. By Taylor expansion of $\nabla\theta(x)$ around x^r we have $(\nabla\theta(x^r+\xi))^T = (\nabla\theta(x^r))^T + H(x^r)\xi + f(\xi)$ where $|f(\xi)| \leqq \beta\|\xi\|^2$ for some positive number β, when ξ is sufficiently close to zero. Assuming that x^r is sufficiently close to $\overline{x}$, and substituting $\xi = \overline{x} - x^r$, we get $0 = (\nabla\theta(\overline{x}))^T = (\nabla\theta(x^r))^T + H(x^r)(\overline{x} - x^r) + f(\overline{x} - x^r)$. By the continuity of $H(x)$, and the hypothesis, when x^r is sufficiently close to $\overline{x}$, $H(x^r)$ is also PD and so $(H(x^r))^{-1}$ exists. Multiplying the above equation on both sides by $(H(x^r))^{-1}$ we get (since $x^{r+1} = x^r - (H(x^r))^{-1}(\nabla\theta(x^r))^T$ in Newton's method) $0 = -(x^{r+1} - x^r) + (\overline{x} - x^r) + (H(x^r))^{-1}f(\overline{x} - x^r) = (\overline{x} - x^{r+1}) + (H(x^r))^{-1}f(\overline{x} - x^r)$. Since $H(\overline{x})$ is PD, when x^r is sufficiently close to $\overline{x}$, there exists a constant δ such that $\|H(x^r)^{-1}\| \leqq \delta$. So from the above equation we conclude that

$$\|(\overline{x} - x^{r+1})\| = \| - (H(x^r))^{-1}f(\overline{x} - x^r)\| \leqq \delta\|f(\overline{x} - x^r)\| \leqq \beta\delta\|\overline{x} - x^r\|^2.$$

Using this inequality for $r = 0$, we conclude that there exists an $\varepsilon > 0$ sufficiently small, such that $\|\overline{x} - x^0\| < \varepsilon$ implies $\|\overline{x} - x^1\| < \nu\|\overline{x} - x^0\| < \nu\varepsilon$ where $\nu < 1$. Repeating this argument we conclude that $\|\overline{x} - x^r\| \to 0$ as $r \to \infty$, that is, the sequence $\{x^r\}$ converges to $\overline{x}$. That the convergence is of second order follows from the above inequality.

$\square$

10.8.5 Modified Newton's Methods

When $H(\theta(x^k))$ is PD, $-(H(\theta(x^k)))^{-1}(\nabla\theta(x^k))^T$ is a descent direction for $\theta(x)$ at x^k, but there is no guarantee that $\theta(x^{k+1}) \leqq \theta(x^k)$ when x^{k+1} is determined by (10.16), because the step length is a constant, 1, independent of the data. The sequence can be made into a descent sequence by modifying Newton's method into Newton's method with line search, in which the direction of search is $\pm y^k$ satisfying (10.15), the sign determined (when $H(\theta(x^k))$ is not PD) so as to ensure that the direction is a descent direction, and any of the step length procedures discussed earlier are used for the line search.

The major difficulty with Newton's method arises when $H(\theta(x^k))$ is not PD. If $H(\theta(x^k))$ is singular, (10.15) may not have a solution, and even if it has a solution, when $H(\theta(x^k))$ is not PD, solutions of (10.15) are not necessarily descent directions, and methods based on using them may not converge. In the case when $H(\theta(x^k))$ is not even PSD, it is possible to modify Newton's method by using directions of negative curvature together with step size procedures such as the second order Armijo step.

One modification suggested to guarantee that the search directions are descent directions is to replace $H(\theta(x^k))$ in (10.15) by $\alpha^k Q^k + H(\theta(x^k))$ where Q^k is either I or a positive diagonal matrix and α^k is a positive number to ensure that the resulting matrix is PD, and then solve the modified equation to give the search direction to be used at x^k.

For other modified versions of Newton's method see [10.9, 10.13, 10.26, A8].

One main difficulty in using Newton's method (or modified Newton methods) is that the Hessian matrix has to be evaluated in each step. If subroutines for directly computing each element of the Hessian matrix are not available, they can be approximated by finite differences of the gradient vector. For this, select a positive number α, the finite difference interval. To approximate the Hessian at the point x^k, compute

$$H_{.i} = \frac{1}{\alpha}(\nabla\theta(x^k + \alpha I_{.i}) - \nabla\theta(x^k))^T.$$

Let H be the matrix with columns $H_{.i}$, $i = 1$ to n. Then $(H + H^T)/2$ can be used as an approximation for $H(\theta(x^k))$ in executing Newton's or the appropriate modified Newton's method. With this change, the method is usually called a **discrete** (or **finite difference**) **Newton** or **modified Newton method**. These methods are very worthwhile when the Hessian matrix has a known sparsity pattern.

10.8.6 Quasi Newton Methods

Newton's method is difficult to implement because of the computational burden involved in calculating the Hessian matrix in each step (even if we decide to use a finite difference approximation for it). The Quasi-Newton methods try to build up information on the Hessian through various steps of the descent method using the computed values of $\nabla\theta(x)$ and $\theta(x)$. In these methods $(H(\theta(x^k)))^{-1}$ is approximated by a symmetric positive definite matrix, D_k, which is updated in each iteration. Thus in these methods, the $(k+1)$th step consists of the following.

(a) Initiate this step with the point x^k obtained in the previous step (if $k = 0$, initiate this step with x^0, some initial point with which the method is started).

(b) Compute the search direction at x^k, denoted by $y^k = -D_k(\nabla\theta(x^k))^T$.

(c) Compute step length in the search direction, $\alpha_k > 0$, by doing a line search, leading to the new point $x^{k+1} = x^k + \alpha_k y^k$.

(d) Check whether termination criteria (see Section 10.8.8) are satisfied by the new point x^{k+1}, in which case accept x^{k+1} as the solution of (10.4) and terminate. Otherwise update D_k giving D_{k+1} and go to the next step.

The methods start out with an initial solution x^0, and a symmetric positive definite matrix D_0 (usually $D_0 = I$). D_k is an approximation to the inverse Hessian at a local minimum to which the sequence of points generated is presumed to converge. Different algorithms use different formula for updating D_k from step to step. The advantages are that these methods only need the computation of the gradient vector $\nabla\theta(x)$ at one point in each step. When the matrices D_k are all PD, the search directions $y_k = -D_k(\nabla\theta(x^k))^T$ are descent directions. In some quasi-Newton methods D_k may not always be PD, but the important methods do maintain this property. When D_k is PD, the search direction y^k is the steepest descent direction at x^k using D_k^{-1} as the metric matrix, and since this metric matrix changes from iteration to iteration, these methods are also known as **variable metric methods**.

The updating formula which gives D_{k+1} as a function of D_k attempts to take into account the second derivative information obtained during the $(k+1)$th step. The formula is derived to ensure that D_k becomes a good approximation of $(H(\theta(x^k)))^{-1}$ as the method progresses. This is done through the use of an equation known as the **quasi-Newton condition**, which we will now derive. By taking the Taylor series expansion of $\nabla\theta(x)$ around the point x^k and neglecting higher order terms, we get

$$(\nabla\theta(x^{k+1}))^T \simeq (\nabla\theta(x^k))^T + H(\theta(x^k))(x^{k+1} - x^k).$$

So, if $H(\theta(x^k))$ is invertible, we have

$$(H(\theta(x^k)))^{-1}(\nabla\theta(x^{k+1}) - \nabla\theta(x^k))^T \simeq (x^{k+1} - x^k). \tag{10.17}$$

Since the quantities x^{k+1} and $\nabla\theta(x^{k+1})$ are not available until the $k+1$th step is completed, we cannot expect the matrix D_k to satisfy (10.17) in place of $(H(\theta(x^k)))^{-1}$, but we could require D_{k+1} to satisfy

$$D_{k+1}(\nabla\theta(x^{k+1}) - \nabla\theta(x^k))^T = (x^{k+1} - x^k). \tag{10.18}$$

This condition is the quasi-Newton condition, and the updating formulae for the matrices D_k in quasi-Newton methods are usually formulated so that this condition holds for all k. If the updating formulae are such that this condition is satisfied, and satisfies certain other prior conditions (sometimes it is also required that

$D_{k+1}(\nabla\theta(x^{j+1}) - \nabla\theta(x^j))^T = (x^{j+1} - x^j)$ hold for all $j \leqq k$) it can be shown that when the algorithm is applied to minimize $\frac{1}{2}x^TAx + cx$ where A is PD and symmetric, using exact line searches, then the search directions generated are conjugate directions (see Section 10.8.7 for the definition of conjugate directions), that $D_n = A^{-1}$, and that the method terminates after at most n steps with the minimum.

The three basic considerations in constructing updating formulae for D_k in quasi-Newton methods are (i) the quasi-Newton condition (10.18), (ii) hereditary symmetry (i. e., if D_k is symmetric, the updating formula should guarantee that D_{k+1} is also symmetric), and (iii) hereditary positive definiteness. Not all the quasi-Newton methods satisfy all these properties. In some of them, these properties may only hold if the line searches are carried out to a high degree of precision in each iteration.

The updating formula usually has the form $D_{k+1} = D_k + C_k$ where C_k is a matrix known as the correction term. Usually C_k has rank 1 or 2, and depending on its rank, the methods are classified either as rank-one or rank-two methods.

Now we will present the updating formulas used by some important quasi-Newton methods. The remaining details are the same as discussed above, for each method. For $k \geqq 1$, we define

$$\begin{aligned} \xi^k &= x^k - x^{k-1} \\ \eta^k &= (\nabla\theta(x^k) - \nabla\theta(x^{k-1}))^T \end{aligned} \tag{10.19}$$

The Davidon-Fletcher-Powell (DFP) Method

Here the updating formula is

$$D_{k+1} = D_k + \frac{\xi^{k+1}(\xi^{k+1})^T}{(\xi^{k+1})^T\eta^{k+1}} - \frac{(D_k\eta^{k+1})(D_k\eta^{k+1})^T}{(\eta^{k+1})^TD_k\eta^{k+1}}$$

where ξ^{k+1}, η^{k+1} are column vectors defined as in (10.19). The method has the hereditary symmetry property. It also has the hereditary PD property if $(\xi^{k+1})^T\eta^{k+1} > 0$ for all k. Notice that this condition will hold if the search direction y^k is a descent direction and the line search is carried out optimally or to a local minimum. The method has superlinear rate of convergence. When applied to minimize a strictly convex quadratic function $\frac{1}{2}x^TAx + cx$ with exact line searches, the method preserves the condition $D_{k+1}\eta^{j+1} = \xi^{j+1}$ for all $j \leqq k$, for all k; it generates conjugate search directions and terminates after n steps with $D_{n+1} = A^{-1}$ and the optimum solution. See [10.9, 10.13, 10.37, A3] for proofs of these results.

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) Method

Here the updating formula is

$$\begin{aligned} D_{k+1} =& D_k + \left(1 + \frac{(\eta^{k+1})^TD_k\eta^{k+1}}{(\xi^{k+1})^T\eta^{k+1}}\right)\left(\frac{\xi^{k+1}(\xi^{k+1})^T}{(\xi^{k+1})^T\eta^{k+1}}\right) - \\ & \frac{\xi^{k+1}(\eta^{k+1})^TD_k + D_k\eta^{k+1}(\xi^{k+1})^T}{(\xi^{k+1})^T\eta^{k+1}} \end{aligned}$$

where ξ^{k+1}, η^{k+1} are column vectors defined as in (10.19). The method has the hereditary symmetry and hereditary PD properties, satisfies the quasi-Newton conditions ($D_{k+1}\eta^{j+1} = \xi^{j+1}$ for all $j \leqq k$), and has the quadratic termination property. At present this is considered the best quasi-Newton method. The method has been shown to converge even with inexact line searches (using several of the line search termination criteria discussed in Section 10.7.8).

Resetting in Quasi-Newton Methods

In quasi-Newton methods, the steps can continue until termination. However, in some implementations the method is reset by setting the matrix D_k to some positive definite matrix (usually the same as D_0, or I) after every n steps. If implemented this way, the method goes through cycles. Each cycle begins with the point obtained at the end of the last step in the previous cycle (the initial cycle begins with the initial point x^0 with which the method is initiated) and the initial step of each cycle begins with the matrix D_0 (usually I or some other PD symmetric matrix) and the cycle consists of n steps.

Also in each step one should check that the search direction y^k satisfies $(\nabla\theta(x^k))y^k < 0$, as otherwise the direction is not a descent direction. Usually the method is also reset whenever this descent condition is violated.

See references [10.2, 10.8, 10.9, 10.13, 10.26, 10.27, 10.33, A3] for a discussion of various other quasi-Newton methods, their best computer implementations, and the convergence results established about them.

10.8.7 Conjugate Direction Methods

These are a class of methods that use only first order derivatives, which obtain search directions without the need for storing or updating a square matrix of order n. Conjugate direction methods were developed with the aim of solving strictly convex quadratic programming problems with an effort of at most n line searches. For this, the search directions have to be chosen to satisfy the conjugancy property. Let $f(x) = cx + \frac{1}{2}x^TAx$ where A is a PD symmetric matrix of order n. Consider the linear transformation $x = Pz$ where P is a nonsingular square matrix of order n. This transforms $f(x)$ into $F(z) = cPz + \frac{1}{2}z^TP^TAPz$. $F(z)$ can be minimized with an effort of at most n line searches in the z-space if it is separable, that is, if $P^TAP = Q$ is a diagonal matrix with positive diagonal entries $Q_{11}, \ldots, Q_{nn}$, and

$$(P_{.i})^TAP_{.j} = 0 \quad \text{for each } i \neq j. \tag{10.20}$$

In this case $F(z)$ is equal to $\sum_{j=1}^{n} F_j(z_j)$ where $F_j(z_j)$ involves only one variable, and hence minimizing $F(z)$ over $z \in \mathbf{R}^n$ can be achieved by n one dimensional problems of minimizing $F_j(z_j)$ over $z_j \in \mathbf{R}^1$ for each $j = 1$ to n separately, that is, n line searches. The set of nonzero vectors $\{P_{.1}, \dots, P_{.n}\}$ is said to be **conjugate with respect to the PD symmetric matrix** A if (10.20) holds. Let $\nu = (\nu_1, \dots, \nu_n) = cP$. So $F(z) = \sum_{j=1}^{n} \nu_j z_j + \frac{1}{2}\sum_{j=1}^{n} Q_{jj} z_j^2$. Hence, the point which minimizes $F(z)$ is $\overline{z} = (\overline{z}_j) = (-\nu_j/Q_{jj})$ and so the point which minimizes $f(x)$ in the x-space is $\overline{x} = (\overline{x}_j) = P\overline{z}$. Since $F(z)$ is separable, we can visualize the minimum of $F(z)$ as being obtained by starting at an arbitrary point z_0 in the z-space and doing n line searches exactly, once in each direction $I_{.j}$, $j = 1$ to n (the alternating variables method). Let z^j be the point obtained at the end of the jth line search in this scheme. So z^{j+1} is the minimizer of $F(z^j + \alpha I_{.j+1})$ over $\alpha \in \mathbf{R}^1$, $j = 0$ to $n-1$. Then $z^n = \overline{z}$. If $x^j = Pz^j$, $j = 0$ to n, it can be verified that x^{j+1} is the minimizer of $f(x^j + \alpha P_{.j+1})$ over $\alpha \in \mathbf{R}^1$, $j = 0$ to $n-1$ and that $x^n = \overline{x}$, the point which minimizes $f(x)$. The following properties can be verified to hold

1. the conjugacy condition (10.20) implies that $\{P_{.1}, \dots, P_{.n}\}$ is linearly independent.
2. $(\nabla f(x^{k+1}))P_{.j} = 0$, for $j = 1$ to k.
3. Let α_j be the minimizer of $f(x^j + \alpha P_{.j+1})$ over $\alpha \in \mathbf{R}^1$, for $j = 0$ to $n-1$. Then $x^{j+1} = x^j + \alpha_j P_{.j+1}$. So $(\nabla f(x^{j+1}) - \nabla f(x^j))^T = A(x^{j+1} - x^j) = \alpha_j AP_{.j+1}$. So $(\nabla f(x^{i+1}) - \nabla f(x^i))P_{.j} = 0$ for $i \neq j$.

The conjugate gradient methods for minimizing $f(x)$ construct the conjugate directions one after the other using information collected from earlier line searches. Each direction will be a descent direction at the point which is the current point in the step in which this direction is generated. We now describe these methods.

Step 1 is initiated with an arbitrary initial point x^0. The search direction in step 1 is the steepest descent one, $y^0 = -(\nabla f(x^0))^T$. Do a line search to minimize $f(x^0 + \alpha y^0)$, $\alpha \geqq 0$.

The general $(k+1)$th step for $k \geqq 1$ begins with the point x^k obtained at the end of the line search in the kth step. The search direction in this step is

$$y^k = -(\nabla f(x^k))^T + \beta_k y^{k-1}$$

where β_k is a scalar. The various conjugate gradient algorithms use different formula for β_k. They are

$$\beta_k = \|\nabla f(x^k)\|^2 / \|\nabla f(x^{k-1})\|^2 \tag{10.21}$$

in Fletcher and Reeves method [10.13].

$$= (\nabla f(x^k) - \nabla f(x^{k-1}))(\nabla f(x^k))^T / \|\nabla f(x^{k-1})\|^2 \tag{10.22}$$

in Polak and Ribiere and Polyak's method [10.13, 10.17, 10.37].

$$= -\|\nabla f(x^k)\|^2 / (\nabla f(x^{k-1}))y^k \tag{10.23}$$

in conjugate descent method [10.13].

It can be verified that $(\nabla f(x^k))y^k = -\|\nabla f(x^k)\|^2$ if the line search in the previous step is carried out exactly, and in this case y^k is therefore a descent direction at x^k. Now do a line search to minimize $f(x^k + \alpha y^k)$, $\alpha \geqq 0$. If α_k is the optimum step length,

$x^{k+1} = x^k + \alpha_k y^k$. If $\nabla f(x^{k+1}) = 0$, x^{k+1} minimizes $f(x)$, terminate. Otherwise, go to the next step.

The method terminates after at most n steps. It can be verified that the search directions generated are conjugate with respect to the Hessian matrix A, and they are all descent directions if the line search is carried out exactly in each step. Since $f(x)$ is quadratic, it can be verified that β_k obtained in (10.21) or (10.22) or (10.23) are exactly the same if all the line searches are carried out exactly.

To solve the problem of minimizing $\theta(x)$, which is in general not quadratic, we apply the method exactly as above, replacing $f(x)$ by $\theta(x)$ wherever it appears. In this general problem, the search directions generated will be descent directions as long as line searches are carried out exactly in each step. In this general problem, the values for β_k obtained from (10.21), (10.22), (10.23) may be different. In numerical experiments the method using (10.22) seemed to perform better, particularly when n is large. The application of the method can be continued until some termination condition is satisfied (see Section 10.8.8). In practical implementations to minimize general non-quadratic functions $\theta(x)$, the method is usually restarted (or reset) after every n steps. If this is done, the method goes through several cycles. Each cycle consists of n steps. Step 1 of each cycle begins with the point obtained at the end of the previous cycle (or x^0, the initial point, for the first cycle) and uses the negative gradient search direction. In the general non-quadratic case, if inexact line searches are used, the directions generated, y^k, may not be descent directions (that is, $(\nabla\theta(x^k))^T y^k$ may not be < 0). The method based on updating using (10.23) (the conjugate descent method) produces descent directions even when line searches are not very exact. If the search direction in a step is not descent, we can carry out the line search in that step over the entire line (instead of the half-line with step length $\alpha \geqq 0$ as is done usually, that is, allow step length to be negative), but usually the cycle is terminated in such a step and the method is reset to begin the next cycle with the steepest descent direction in step 1. It can be shown that these methods have superlinear convergence in terms of cycles. See [10.8, 10.13, 10.17, 10.26, 10.37].

10.8.8 Practical Termination Conditions for Unconstrained Minimization Algorithms

When the descent algorithm generates the sequence of points $\{x^r : r = 0, 1, \ldots\}$ in practical implementations for minimizing $\theta(x)$, the method can be terminated when some or all of the following conditions are met

$$\begin{aligned} |\theta(x^k) - \theta(x^{k-1})| &< \varepsilon_1 \\ \|x^k - x^{k-1}\| &< \varepsilon_2 \\ \|\nabla\theta(x^k)\| &< \varepsilon_3 \end{aligned}$$

where the ε's are suitably chosen tolerances.

10.9 SURVEY OF SOME METHODS FOR LINEAR EQUALITY CONSTRAINED MINIMIZATION IN $\mathbf{R}^n$

Here we consider the NLP

$$\begin{array}{ll} \text{minimize} & \theta(x) \\ \text{subject to} & Ax = b \end{array} \tag{10.24}$$

where A is a matrix of order $m \times n$ and rank m, and $\theta(x)$ is a real valued continuously differentiable function. Given a feasible point x for this problem, the first order necessary conditions for it to be a local minimum are that there exist a Lagrange multiplier vector $\pi = (\pi_1, \ldots, \pi_m)$ satisfying

$$(\nabla\theta(x)) = \pi A. \tag{10.25}$$

Suppose (10.24) is feasible, and let x be any feasible solution for it. Then every feasible solution for (10.24) is of the form $\tilde{x} + z$ where z satisfies

$$Az = 0. \tag{10.26}$$

There exists a matrix Z of order $n \times (n-m)$ and rank $n-m$, such that every column vector of Z is a solution of (10.26) and conversely every solution of (10.26) is a linear combination of the column vectors of Z. To obtain a matrix like Z, find a basis B for (10.24). B is a square nonsingular submatrix of A of order m. Rearrange the variables and their columns in A so that A can be partitioned into basic and nonbasic parts as (B, D) where D is the $m \times (n-m)$ matrix of nonbasic columns. Then the matrix Z can be taken to be

$$Z = \begin{pmatrix} -B^{-1}D \\ I_{n-m} \end{pmatrix} \tag{10.27}$$

where I_{n-m} is the unit matrix of order $n-m$. It is not necessary to compute Z explicitly. All the computations in the algorithms discussed below can be carried out using a factorization for B^{-1}.

Since any solution for (10.24) is of the form $x = \tilde{x} + Z\xi$ where $\tilde{x}$ is a solution of (10.24) and $\xi \in \mathbf{R}^{n-m}$, (10.24) is equivalent to the problem of minimizing $f(\xi) = \theta(\tilde{x} + Z\xi)$ over $\xi \in \mathbf{R}^{n-m}$, that is the unconstrained minimum of $f(\xi)$ over $\xi \in \mathbf{R}^{n-m}$. It can be verified that $\nabla f(\xi) = (\nabla_x \theta(\tilde{x} + Z\xi))Z$. Also if $\theta(x)$ is twice continuously differentiable, $H(f(\xi)) = Z^T H_x(\theta(\tilde{x} + Z\xi))Z$. For x feasible to (10.24) the vector $(\nabla\theta(x))Z$ is known as the **projected gradient** or the **reduced gradient vector of** $\theta(x)$ at x, and the matrix $Z^T H(\theta(x))Z$ of order $(n-m) \times (n-m)$ is known as the **reduced** or **projected Hessian matrix of** $\theta(x)$ at x. The condition (10.25) implies

$$(\nabla\theta(x))Z = 0. \tag{10.28}$$

If $\theta(x)$ is twice continuously differentiable, a second order necessary condition for the feasible solution x of (10.24) to be a local minimum for it is that the matrix $Z^T H(\theta(x))Z$ is PSD.

The algorithms discussed in this section generate a sequence of feasible points $\{x^0, x^1, \dots\}$ beginning with the initial feasible point x^0. If x^k is feasible, the search direction at x^k in step $k+1$ must satisfy $Ay^k = 0$, that is, $y^k = Z\xi^k$ for some $\xi^k \in \mathbf{R}^{n-m}$, such directions are called feasible search directions, because a move of any length in such a direction, starting from a feasible point, remains in the feasible region for (10.24). Step $k+1$ of the algorithm consists of the following tasks:

1. Compute a feasible search direction: First compute ξ^k and then compute the search direction $y^k = Z\xi^k$.
2. Determine step length: Compute the positive step length α_k.
3. Compute the new point $x^{k+1} = x^k + \alpha_k y^k$.
4. Check whether x^{k+1} satisfies the conditions for termination, if so, accept x^{k+1} as the solution of (10.24) and terminate. Otherwise go to the next step.

The feasible search direction y^k selected in 1. above is a descent direction at x^k if

$$((\nabla\theta(x^k))Z)\xi^k = (\nabla\theta(x^k))y^k < 0. \tag{10.29}$$

The method of steepest descent uses $(\xi^k)^T = -(\nabla\theta(x^k))Z$ to determine the feasible search direction at x^k, which is therefore $y^k = -ZZ^T(\nabla\theta(x^k))^T$, and uses step length procedures exactly as in the unconstrained case. However, this method has slow linear rate of convergence.

Newton's method is based on minimizing the second order Taylor approximation for $f(\xi) = \theta(x^k + Z\xi)$ around $\xi = 0$, that is $\theta(x^k) + (\nabla\theta(x^k))Z\xi + \frac{1}{2}\xi^T Z^T H(\theta(x^k))Z\xi$. So, Newton's method uses the search direction $y^k = Z\xi^k$, where ξ^k solves

$$(Z^T H(\theta(x^k))Z)\xi = -Z^T(\nabla\theta(x^k))^T \tag{10.30}$$

and uses fixed step lengths of $\alpha_k = 1$. Modified Newton methods replace the matrix $Z^T H(\theta(x^k))Z$ in (10.30) (when this matrix is not PD) by a PD approximation to it such as $Z^T H(\theta(x^k))Z + \nu I$ for some $\nu > 0$, and step lengths determined by line searches.

When the second derivatives are not available, the matrix $Z^T H(\theta(x^k))Z$ can be approximated by finite difference approximation. For this, let ε_i be an appropriate finite difference interval, and for $i = 1$ to $n-m$ let

$$W_{.i} = \frac{1}{\varepsilon_i}(\nabla\theta(x^k + \varepsilon_i Z_{.i}) - \nabla\theta(x^k))^T$$

and let W be the $n \times (n-m)$ matrix with column vectors $W_{.i}$, $i = 1$ to $n-m$. Then a symmetric approximation for $Z^T H(\theta(x^k))Z$ is $(1/2)(Z^T W + W^T Z)$.

Quasi-Newton methods can be developed for (10.24) by looking at the corresponding unconstrained minimization problem of minimizing $f(\xi) = \theta(x^k + Z\xi)$, but carrying out all the operations in the x-space. In this case the search direction in step $k+1$ will be $y^k = Z\xi^k$, where $\xi^k = -D_k Z^T(\nabla\theta(x^k))^T$. The matrix D_k is of order $(n-m)\times(n-m)$. We choose $D_0 = I_{n-m}$, and in updating D_k from step to step, we use the updating formulas discussed in Section 10.8.6 with $\xi^k = Z^T(x^{k+1}-x^k)$, and $\eta^k = Z^T(\nabla\theta(x^{k+1}) - \nabla\theta(x^k))^T$ instead of (10.19).

Another approach for solving (10.24) is to use a conjugate gradient method on the corresponding reduced problem of minimizing $f(\xi) = \theta(x^k + Z\xi)$, but doing all the computations in the x-space. The search directions used are

$$\begin{aligned} y^1 &= -Z(\nabla\theta(x^0)Z)^T \\ y^k &= -Z(\nabla\theta(x^k)Z)^T + \beta_k y^{k-1} \end{aligned}$$

where $\beta_k = \|(\nabla\theta(x^k))Z\|^2/\|(\nabla\theta(x^{k-1}))Z\|^2$ or $(\nabla\theta(x^k) - \nabla\theta(x^{k-1}))ZZ^T(\nabla\theta(x^k))^T/\|\nabla\theta(x^{k-1})Z\|^2$, or $-\|\nabla\theta(x^k)Z\|^2/(\nabla\theta(x^{k-1})Z)\xi^k$ (here ξ^k is the unique solution of $Z\xi^k = y^k$), as in (10.21), (10.22), (10.23), depending on the method used. Statements made in Section 10.8.7 about resetting the algorithm remain valid here also (here resetting is done after every $n-m$ steps or whenever the search direction generated is not a descent direction).

10.9.1 Computing the Lagrange Multiplier Vector

Let $\overline{x}$ be the terminal point obtained in the algorithm for solving (10.24). The corresponding Lagrange multiplier vector is the vector $\overline{\pi}$ which satisfies (10.25). Given $\overline{x}$, (10.25) is a system of n equations in the m unknowns $\pi_1,\ldots,\pi_m$, and since $n > m$, this is an overdetermined system of equations. We can determine $\overline{\pi}$ as the row vector in $\mathbf{R}^m$ which minimizes $\|(\nabla\theta(\overline{x}))^T - \pi A\|^2$ over $\pi \in \mathbf{R}^m$, for which the solution is given by

$$\overline{\pi} = (AA^T)^{-1}A\nabla\theta(\overline{x}). \tag{10.31}$$

If $\overline{x}$ is a local minimum for (10.24), the vector $\overline{\pi}$ given by (10.31) is an exact solution for (10.25). If $\overline{x}$ is an approximation to a local minimum (obtained when the algorithms discussed above are terminated using some practical termination criteria discussed in Section 10.8, there is no π satisfying (10.25) exactly, however, the $\overline{\pi}$ obtained from (10.31) is a corresponding approximation to the Lagrange multiplier vector for (10.24). For other approximating estimates to the Lagrange multiplier vector see references [10.13, 10.17].

10.10 SURVEY OF OPTIMIZATION SUBJECT TO GENERAL LINEAR CONSTRAINTS

10.10.1 The Use of Lagrange Multipliers to Identify Active Inequality Constraints

For the purpose of this discussion, consider the following NLP:

$$\begin{array}{ll} \text{minimize} & \theta(x) \\ \text{subject to} & Ax \geqq b \end{array} \tag{10.32}$$

where A is a matrix of order $m \times n$, say. If $\overline{x}$ is feasible, the ith constraint in (10.32) is said to be active or tight or binding at $\overline{x}$ if it holds as an equation at $\overline{x}$, that is, if $A_{i.}\overline{x} = b_i$; inactive if $A_{i.}\overline{x} > b_i$. For $\overline{x}$ feasible to (10.32), let $\mathbf{I}(\overline{x}) = \{i : i \text{ such that } A_{i.}\overline{x} = b_i\}$ = index set of active constraints in (10.32) at $\overline{x}$. Let $y \in \mathbf{R}^n$, $y \neq 0$. y is said to be a feasible direction at $\overline{x}$, if $\overline{x} + \lambda y$ remains feasible for (10.32) for all $0 \leqq \lambda \leqq \overline{\lambda}$, for some positive $\overline{\lambda}$. Clearly y is a feasible direction at $\overline{x}$ iff

$$A_{i.}y \geqq 0, \quad \text{for each } i \in \mathbf{I}(\overline{x}). \tag{10.33}$$

The direction y is said to be a **binding direction** or a **non-binding direction** at $\overline{x}$ with respect to the ith constraint for $i \in \mathbf{I}(\overline{x})$, depending on whether $A_{i.}y = 0$ or $A_{i.}y > 0$ respectively. A move in a binding direction continues to keep the constraint active, while any move of positive length in a non-binding direction makes the constraint inactive, that is, moves off the constraint.

Now consider the corresponding equality constrained NLP:

$$\begin{array}{ll} \text{minimize} & \theta(x) \\ \text{subject to} & Ax = b \end{array} \tag{10.34}$$

and further assume that the set of row vectors of A is linearly independent. Suppose $\overline{x}$ is a KKT point for (10.34) with the associated Lagrange multiplier vector $\overline{\pi} = (\overline{\pi}_1, \ldots, \overline{\pi}_m)$. So $\overline{x}$, $\overline{\pi}$ together satisfy the first order necessary optimality conditions

$$\nabla\theta(\overline{x}) = \overline{\pi}A. \tag{10.35}$$

Since the set of feasible solutions of (10.34) is a subset of the set of feasible solutions of (10.32), an optimum solution for (10.34) may not be optimal for (10.32) in general. The point $\overline{x}$ is of course feasible to (10.32) and clearly it is also a KKT point for (10.32) if $\overline{\pi} \geqq 0$.

Suppose there is a t such that $\overline{\pi}_t < 0$, we will now show that there exists a descent feasible direction at $\overline{x}$ for (10.32) which moves off the tth constraint. Since the set of row vectors of A is assumed to be linearly independent, by standard results in linear algebra, there exists a $y \in \mathbf{R}^n$ satisfying

$$\begin{aligned} A_{i.}y &= 1 \quad \text{for } i = t \\ &= 0 \quad \text{for } i \neq t. \end{aligned} \tag{10.36}$$

Let y be a solution for (10.36). From (10.35) and (10.36), we have $(\nabla\theta(\overline{x}))y = \overline{\pi}Ay = \overline{\pi}_t < 0$, and hence y is a descent feasible direction for (10.32) at $\overline{x}$.

Thus a necessary condition for a KKT point of (10.34) to be a KKT point for (10.32) is that all the Lagrange multipliers be nonnegative. Otherwise we can construct a descent feasible direction for (10.32) at such a point. These results are used in some of the algorithms discussed below, to solve NLP's involving linear inequality constraints using techniques for solving NLP's involving linear equality constraints only. They try to guess the set of active inequality constraints at the optimum, and apply the equality constraint techniques to the problem treating these active constraints as equations. Modifications are made in the active set using Lagrange mulitplier information gathered over each step.

10.10.2 The General Problem

Here we consider the NLP

$$\begin{array}{lll} \text{minimize} & \theta(x) & \\ \text{subject to} & A_{i.}x = & b_i, \ i = 1 \text{ to } m \\ & \phantom{A_{i.}x} \geqq & b_i, \ i = m+1 \text{ to } m+p \end{array} \tag{10.37}$$

where $x \in \mathbf{R}^n$, and $\theta(x)$ is a real valued continuously differentiable function. Given a feasible point x, the first order necessary conditions for x to be a local minimum for this problem are that there exists a Lagrange multiplier vector $\pi = (\pi_1, \ldots, \pi_{m+p})$ satisfying

$$\begin{aligned} \nabla\theta(x) &= \sum_{i=1}^{m+p} \pi_i A_{i.} \\ \pi_i &\geqq 0, \ i = m+1 \text{ to } m+p \\ \pi_i(A_{i.}x - b_i) &= 0, \ i = m+1 \text{ to } m+p. \end{aligned} \tag{10.38}$$

Without any loss of generality we assume that $\{A_{i.} : i = 1 \text{ to } m\}$ is linearly independent. Let $\mathbf{K}$ denote the set of feasible solutions of (10.37). Given $\overline{x} \in \mathbf{K}$, all the equality constraints for $i = 1$ to m are active at $\overline{x}$ in (10.37). For $m+1 \leqq i \leqq m+p$, the ith constraint in (10.37) is active at $\overline{x}$ (also said to be an active inequality constraint at $\overline{x}$) if $A_{i.}\overline{x} = b_i$, inactive otherwise. Let $\mathbf{I}(\overline{x}) = \{i : A_{i.}\overline{x} = b_i\}$, the index set of active constraints at $\overline{x}$. The point $y \in \mathbf{R}^n$, $y \neq 0$, is a feasible direction at $\overline{x}$ if $\overline{x} + \lambda y \in \mathbf{K}$ for $0 \leqq \lambda \leqq \overline{\lambda}$, for some positive $\overline{\lambda}$. Clearly y is a feasible direction at $\overline{x}$ iff

$$\begin{aligned} A_{i.}y &= 0, \ i = 1 \text{ to } m \\ &\geqq 0, \ i \in \mathbf{I}(\overline{x}) \cap \{m+1, \ldots, m+p\}. \end{aligned}$$

If y is a feasible direction at $\overline{x}$ and $A_{i.}y > 0$ for some $i \in \mathbf{I}(\overline{x}) \cap \{m+1, \ldots, m+p\}$, a move in the direction y from $\overline{x}$ is said to move off the ith constraint in (10.37).

We will now discuss some algorithms for solving (10.37).

10.10.3 The Frank-Wolfe Method

To solve (10.37), this method generates a descent sequence of feasible points $\{x^r : r = 0, 1, \ldots\}$ beginning with an initial feasible solution x^0, satisfying $\theta(x^{r+1}) < \theta(x^r)$ for all r.

For $k \geqq 0$, in step $k+1$, the initial point is x^k, the feasible point obtained at the end of the previous step if $k > 0$, or the feasible point with which the method is initiated, if $k = 0$. In this step the search direction y^k is of the form $z^k - x^k$ where z^k is a feasible point satisfying $(\nabla\theta(x^k))(z^k - x^k) < 0$, and so y^k is a descent direction at x^k. To find a point like z^k, we solve the LP in variables x

$$\begin{array}{ll} \text{minimize} & (\nabla\theta(x^k))x \\ \text{subject to} & x \in \mathbf{K}. \end{array} \tag{10.39}$$

If z^k is an optimum solution obtained when the LP is solved and $(\nabla\theta(x^k))^T z^k = (\nabla\theta(x^k))^T x^k$, then x^k is also optimal to the LP (10.39). By the duality theorem of linear programming, there exists a vector π^k such that x^k, π^k together satisfy the first order necessary optimality conditions (10.38) for (10.37), and so we terminate with x^k as the solution for (10.37). Otherwise, since $x^k \in \mathbf{K}$, we must have $(\nabla\theta(x^k))(z^k - x^k) < 0$, and so $y^k = z^k - x^k$ is a feasible descent direction at x^k. Now do a line search to find the minimum of $\theta(x^k + \alpha y^k)$ subject to $0 \leqq \alpha \leqq 1$. If α_k is the minimum for this line search problem, the next point in the sequence is $x^{k+1} = x^k + \alpha_k y^k$, continue.

We have the following results about the convergence properties of this method.

Theorem 10.2 *Suppose $\mathbf{K} \neq \emptyset$ and that the linear function in x, $(\nabla\theta(\tilde{x}))x$, is bounded below on $x \in \mathbf{K}$ for each $\tilde{x} \in \mathbf{K}$. Assume that $\mathbf{K}$ has at least one extreme point, and that for each k, the optimum solution z^k for the LP (10.39) obtained in the method is an extreme point of $\mathbf{K}$. If the method does not terminate after a finite number of steps, the sequence $\{x^r : r = 0, 1, \ldots\}$ generated by the above method has at least one limit point, and every limit point of this sequence is a KKT point for (10.37), if the line searches are carried out exactly in each step.*

Proof. Since $\nabla\theta(\tilde{x})x$ is bounded below for $x \in \mathbf{K}$ for each $\tilde{x} \in \mathbf{K}$, the LP (10.39) has an optimum solution always. The LP (10.39) may have alternate optima, and we are assuming that z^k is an optimum solution for (10.39) which is an extreme point of $\mathbf{K}$ (this will be the case, for example, if $\mathbf{K}$ has at least one extreme point and (10.39) is solved by the simplex method). Since $\mathbf{K}$ is a convex polyhedron, it has a finite number of extreme points, and let $\mathbf{K}^{\Delta}$ be the convex hull of these extreme points. Because of the descent property $\theta(x^r)$ is monotonic decreasing as r increases, and by the manner in which the algorithm is carried out, it is clear that every point in the infinite sequence $\{x^r\}$ lies in the convex hull of $\mathbf{K}^{\Delta}$ and x^0, a compact set. So the sequence $\{x^r\}$ has at least one limit point. Let $\overline{x}$ be a limit point of the sequence $\{x^r\}$. Let $\mathbf{S}$ be an infinite set of positive integers such that $x^k \to \overline{x}$ as $k \to \infty$ with all $k \in \mathbf{S}$. For each $k \in \mathbf{S}$ we have an associated extreme point of $\mathbf{K}$, z^k, which is an optimum solution of (10.39). Since there are only a finite number of extreme points of $\mathbf{K}$, there must exist at least one extreme point of $\mathbf{K}$, say $\overline{z}$, which is equal to z^k for $k \in \mathbf{S}$ an infinite number of times. Let $\mathbf{S}_1 \subset \mathbf{S}$ such that for each $k \in \mathbf{S}_1$, $z^k = \overline{z}$. So $(\nabla\theta(x^k))^T(\overline{z} - x^k) < 0$ for each $k \in \mathbf{S}_1$. $x^k \to \overline{x}$ as $k \to \infty$ through $k \in \mathbf{S}_1$, so taking the limit in the above inequality as $k \to \infty$ through $k \in \mathbf{S}_1$, we get

$$(\nabla\theta(\overline{x}))(\overline{z} - \overline{x}) \leqq 0. \tag{10.40}$$

By our hypothesis, the line searches are carried out exactly in each step. Let $\mathbf{S}_1 = \{r_t : t = 1 \text{ to } \infty\}$, with the elements in $\mathbf{S}_1$ arranged in increasing order. So limit $x^{r_t} = \overline{x}$ as $t \to \infty$. In step $k = 1 + r_t$, the optimal step length is α_{1+r_t}, and so we must have, for $0 \leqq \alpha \leqq 1$,

$$\theta(x^{r_t} + \alpha(\overline{z} - x^{r_t})) \geqq \theta(x^{1+r_t}) \geqq \theta(x^{r_{t+1}}). \tag{10.41}$$

This follows because x^{1+r_t} is the point on the line segment $\{x^{r_t} + \alpha(\overline{z} - x^{r_t}) : 0 \leqq \alpha \leqq 1\}$ which minimizes $\theta(x)$ on this line segment. Also, since r_t is an increasing sequence, we have $r_{t+1} \geqq 1 + r_t$, and since $\{\theta(x^1), \theta(x^2), \ldots\}$ is a descent sequence we have $\theta(x^{1+r_t}) \geqq \theta(x^{r_{t+1}})$. In (10.41) let $t \to \infty$. This leads to

$$\theta(\overline{x} + \alpha(\overline{z} - \overline{x})) - \theta(\overline{x}) \geqq 0 \tag{10.42}$$

for all $0 \leqq \alpha \leqq 1$. When α is sufficiently small and positive, by the mean value theorem of calculus, (10.42) implies that $\alpha(\nabla\theta(\overline{x}))(\overline{z} - \overline{x}) \geqq 0$, that is, $(\nabla\theta(\overline{x}))(\overline{z} - \overline{x}) \geqq 0$. Combining this with (10.40) we have

$$\nabla\theta(\overline{x})(\overline{z} - \overline{x}) = 0. \tag{10.43}$$

Since $\overline{z}$ is an optimum solution of (10.39) whenever $k \in \mathbf{S}_1$, and since $x^k \to \overline{x}$ as $k \to \infty$ with all $k \in \mathbf{S}_1$, by (10.43) we conclude that $\overline{x}$ is a feasible solution for (10.37) satisfying the property that $x = \overline{x}$ is an optimum solution of the LP

$$\begin{array}{ll} \text{minimize} & (\nabla\theta(\overline{x}))x \\ \text{subject to} & A_{i.}x = b_i, \; i = 1 \text{ to } m \\ & \qquad \geqq b_i, \; i = m+1 \text{ to } m+p. \end{array} \tag{10.44}$$

Let $\overline{\pi} = (\overline{\pi}_1, \ldots, \overline{\pi}_{m+p})$ be an optimum dual solution associated with (10.44), then by the duality and complementary slackness theorems of linear programming, $\overline{x}$, $\overline{\pi}$ together satisfy (10.38), and hence $\overline{x}$ is a KKT point for (10.37).

$\square$

If $\theta(x)$ is convex, and x^k is a point obtained during the Frank-Wolfe method, and satisfies $(\nabla\theta(x^k))(x^k - z^k) \leqq \varepsilon$, where z^k is an optimum solution of (10.39), then $\theta(x^k) \leqq \varepsilon +$ minimum value of $\theta(x)$ in (10.37). To see this, since $\theta(x)$ is convex, we have for $x \in \mathbf{K}$, $\theta(x) - \theta(x^k) \geqq (\nabla\theta(x^k))(x - x^k) \geqq (\nabla\theta(x^k))(z^k - x^k) \geqq -\varepsilon$, and so $\theta(x) \geqq \theta(x^k) - \varepsilon$ for all $x \in \mathbf{K}$. So if $\theta(x)$ is convex and x^k satisfies $(\nabla\theta(x^k))(x^k - z^k) < \varepsilon$, where ε is small, we can conclude that x^k is near optimum and terminate.

In each step of this method, an LP and a line search problem have to be solved. Even though the system of constraints in the LP to be solved in all the steps is the same, the objective function changes from step to step. The line search problem in each step has to be solved either optimally or at least to guarantee a sufficient decrease in the objective value. Since there is a considerable amount of work to be done in each step, the method tends to be slow. It is practical to use the method only on such problems for which the structure of the problem allows the solution of the LP in each step by an efficient special algorithm. One such application arises in the study of traffic flow along a city's street network using a traffic assignment model. We discuss this application briefly here.

The Traffic Assignment Problem

Let $G = (\mathcal{N}, \mathcal{A})$ be a city's street network. $\mathcal{N}$ is a set of points which are the various centers in the city or street intersections. $\mathcal{A}$ is a set of arcs or street segments, each arc joining a pair of points. The prupose of the study is to determine how the traffic will be distributed over alternate routes. Each driver makes his own choice of the route to take, but traffic flow on road network exhibits certain patterns. One broad principle for the analysis of traffic movement enunciates that traffic distributes itself over alternative routes so that the average journey time is a minimum.

The cost associated with an arc (i, j) in the network is a measure of the journey time from node i to node j along that arc. Journey time is influenced by traffic congestion, and tends to increase with traffic flow. Let f_{ij} denote the traffic flow on this arc (i. e., the number of cars entering this arc at node i per unit time) and let $c_{ij}(f_{ij})$ denote the journey time as a function of the flow f_{ij}. This function has the shape given in Figure 10.10, and so is a monotone increasing convex function.

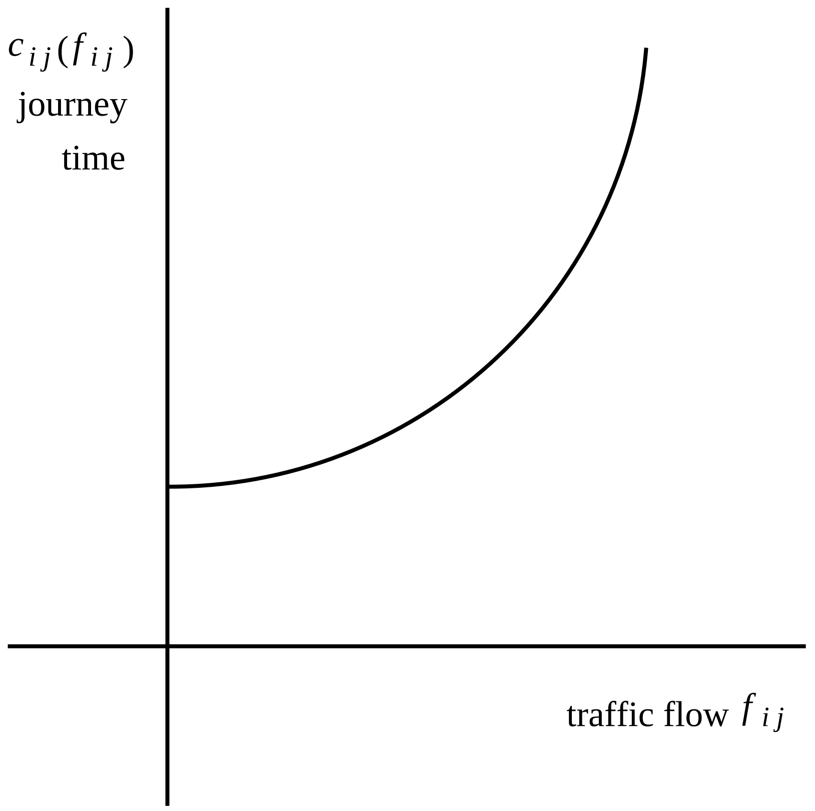

Figure 10.10

Traffic modelers construct these functions $c_{ij}(f_{ij})$ by actually collecting data. They also have data on the volumes of traffic (i. e., how many people travel and from where, to where) for different periods of the day. For example, during a particular peak period, suppose we know that V^u vehicles will be travelling from node s_u (origin) to node t_u (destination) in the network, $u = 1$ to g. Let f^u_{ij} be the number of these vehicles (with origin s_u and destination t_u) travelling along arc (i, j) in the network. For $u = 1$ to g let $f^u = (f^u_{ij})$ be the vector of arc flows of the s_u to t_u vehicle flows. The problem is to determine these vectors f^u. The traffic assignment model states that the $(f^u : u = 1 \text{ to } g)$ form an optimum solution to the following nonlinear (flow dependent cost) multicommodity flow problem

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in\mathcal{A}} c_{ij}(f_{ij}) \\
\text{subject to } f_{ij} = \text{ flow on arc } (i,j) = & \sum_{u=1}^{g} f^u_{ij} \\
\sum(f^u_{ij} : j \text{ such that } (i,j)\in\mathcal{A}) - & \sum(f^u_{ji} : j \text{ such that } (j,i)\in\mathcal{A}) \\
= & \ \ 0, \text{ if } i \neq s_u \text{ or } t_u \\
= & \ \ V^u, \text{ if } i = s_u \\
= & -V^u, \text{ if } i = t_u \\
f^u_{ij} \geqq & \ \ 0, u = 1 \text{ to } g, (i,j)\in\mathcal{A}.
\end{aligned} \tag{10.45}$$

In this model it is quite reasonable to make the simplifying assumption that the flow variables are continuous variables rather than discrete integer variables. Also, since the cost function $c_{ij}(f_{ij})$ is constructed to reflect the journey time as a function of the flow f_{ij}, there is no need to include a constraint in the model corresponding to the capacity for flow of this arc. So, (10.45) is an uncapacitated, convex, multicommodity flow problem, and this can be solved efficiently using the Frank-Wolfe method. It begins with a feasible flow $((f^u)^0 : u = 1 \text{ to } g)$, which can be generated by standard network flow methods, and generates a sequence of feasible flow vectors $((f^u)^r : u = 1 \text{ to } g : r = 0, 1, \ldots)$ converging to the optimum solution of (10.45). In the $(k+1)$th step of this method, the initial flow vectors are $((f^u)^k, u = 1 \text{ to } g)$. Let $(f_{ij})^k = \sum_{u=1}^{g} (f_{ij}^u)^k$, the total flow on arc (i, j) in these flow vectors. Let $\overline{c}_{ij} = \big(\frac{dc_{ij}(f_{ij})}{df_{ij}}$: evaluated at $f_{ij} = (f_{ij})^k\big)$. Then the LP to be solved in this step is

$$\begin{aligned}
\text{minimize} \sum_{u=1}^{g} \quad & \sum_{(i,j)\in\mathcal{A}} \overline{c}_{ij} f_{ij}^u \\
\text{subject to} \sum (f_{ij}^u &: j \text{ such that } (i,j) \in \mathcal{A}) - \sum (f_{ji}^u : j \text{ such that } (j,i) \in \mathcal{A}) \\
&= \ \ 0, \text{ if } i \neq s_u \text{ or } t_u \\
&= \ \ V^u, \text{ if } i = s_u \\
&= -V^u, \text{ if } i = t_u \\
f_{ij}^u &\geqq 0, u = 1 \text{ to } g, (i,j) \in \mathcal{A}.
\end{aligned} \tag{10.46}$$

Clearly, (10.46) can be broken up into g separate network flow problems one for each $u = 1$ to g. Also, the uth problem becomes the shortest chain problem from s_u to t_u in the network $G = (\mathcal{N}, \mathcal{A})$ with $(\overline{c}_{ij})$ as the vector of arc lengths, for which there are very efficient special algorithms.

Let $\mathcal{P}_u$ be the shortest chain from s_u to t_u in G with $(\overline{c}_{ij})$ as the vector of arc costs. Define the flow vector $z^u = (z_{ij})^u$ where

$$\begin{aligned}
z_{ij}^u &= V^u \text{ if } (i,j) \text{ is on } \mathcal{P}_u \\
&= 0 \text{ otherwise.}
\end{aligned}$$

Then the flow vectors $(z^u : u = 1 \text{ to } g)$ are an optimum solution of the LP (10.46), to be solved in this step.

Since the objective function in (10.45) is separable in the arcs, even the line search problem to be solved in this step, which is that of minimizing $\sum_{(i,j)\in\mathcal{A}} c_{ij}(f_{ij}^1 + \ldots + f_{ij}^g)$ over the line segment $\{f^u = \alpha(f^u)^k + (1-\alpha)z^u, u = 1 \text{ to } g, 0 \leqq \alpha \leqq 1\}$, can be simplified.

Thus the Frank-Wolfe method provides a reasonable approach for solving the traffic assignment problem (10.45). The main reason for this is the fact that the LP to be solved in each step of the method breaks down into g separate shortest chain problems, for which very efficient special algorithms are available.

10.10.4 Reduced Gradient Methods

The name reduced gradient method refers to a method which uses the equality constraints to eliminate some variables (the dependent or basic variables) from the problem, and treats the remaining problem in the space of the independent (or nonbasic) variables only, either explicitly or implicitly. The gradient of the objective function in the space of independent variables is the reduced gradient defined in Section 10.9, the search direction is usually the steepest descent vector in the space of the independent variables (the negative reduced gradient vector); or the Newton search direction in the space of the independent variables, determined using the reduced Hessian or an approximation for it.

We will consider the problem in the following form

$$\begin{array}{ll} \text{minimize} & \theta(x) \\ \text{subject to} & Ax = b \\ & l \leqq x \leqq u \end{array} \tag{10.47}$$

where A is a matrix of order $m \times n$ and rank m. As discussed in Chapter 1, the problem (10.37) can be put in this form. Here l, u are the lower and upper bound vectors for x in (10.47). Let B be a basis for A (i. e., a square nonsingular submatrix of A of order m) and partition A as (B, D), and let $x = (x_B, x_D)$ be the corresponding partition of the vector x. x_D is the vector of independent (nonbasic) variables and x_B is the vector of dependent (basic) variables. Let $\overline{x} = (\overline{x}_B, \overline{x}_D)$ be a feasible solution for (10.47). So $\overline{x}_B = B^{-1}(b - D\overline{x}_D)$. The problem can be transformed into one in the space of independent variables x_D only, by eliminating the dependent variables x_B. The reduced gradient at $\overline{x}$ is $\overline{c}_D = (\nabla_{x_D}\theta(\overline{x})) - (\nabla_{x_B}\theta(\overline{x}))B^{-1}D$. Define $\overline{y}_D = (\overline{y}_j)$ by

$$\begin{aligned} \overline{y}_j = -\overline{c}_j & \text{ if } x_j \text{is a nonbasic variable in } x_D \text{ and either } \overline{c}_j < 0 \text{ and} \\ & \overline{x}_j < u_j \text{ or } \overline{c}_j > 0 \text{ and } \overline{x}_j > l_j \\ & = 0 \text{ if } x_j \text{ is a nonbasic variable in } x_D, \\ & \quad \text{and the above conditions not met.} \end{aligned}$$

If $\overline{y}_D = 0$, $\overline{x}$ satisfies the first order necessary optimality condition for being a local minimum for (10.47), and the method terminates. Otherwise verify that $\overline{c}_D\overline{y}_D < 0$, so $\overline{y}_D$ is a descent direction in the space of independent variables x_D. It is the steepest descent (negative reduced gradient) direction. Define $\overline{y}_B = -B^{-1}D\overline{y}_D$ and let $\overline{y} = (\overline{y}_B, \overline{y}_D)$. Then $\overline{y}$ is the search direction at $\overline{x}$. Since $A\overline{y} = 0$, the equality constraints in (10.47) continue to be satisfied when we move in this direction. Define

$$\begin{aligned} \lambda_1 &= \text{ minimum } \{(\overline{x}_j - l_j)/(-\overline{y}_j) : j \text{ such that } \overline{y}_j < 0\}, \\ \lambda_2 &= \text{ minimum } \{(u_j - \overline{x}_j)/(\overline{y}_j) : j \text{ such that } \overline{y}_j > 0\}, \\ \overline{\lambda} &= \text{ minimum } \{\lambda_1, \lambda_2\}. \end{aligned}$$

Do a line search for minimizing $\theta(\overline{x}+\lambda\overline{y})$ over $0 \leqq \lambda \leqq \overline{\lambda}$, and repeat the whole process with the optimum point in this line segment.

Let l_B, u_B denote the bound vectors for the dependent variables x_B. If $l_B < \overline{x}_B < u_B$, from the definition of the search direction $\overline{y}$, it can be verified that $\overline{\lambda} > 0$. If however, (10.47) is degenerate, given a feasible solution $\overline{x}$ for it, it may not be possible to find a basis B for (10.47) for which $l_B < \overline{x}_B < u_B$ holds. In this degenerate case, it may so happen that $\overline{\lambda} = 0$. In this case $\overline{y}$ is not a feasible direction at $\overline{x}$, and the line search problem does not make any sense, since any move of positive length in the direction $\overline{y}$ results in infeasibility. In this case the method can be continued by identifying the active constraints at $\overline{x}$, and moving from $\overline{x}$ in the direction of $\overline{y}^p$, the orthogonal projection of $\overline{y}$ in the subspace of active constraints at $\overline{x}$ (this will be a gradient projection step, see the next section, Section 10.10.5). This is equivalent to carrying out the line search problem exactly as above after replacing $\overline{y}$ by $\overline{y}^p$.

For convergence and rate of convergence results in this method see [10.2, 10.13, 10.15, 10.17, 10.26].

This method has been generalized very directly into the Generalized Reduced Gradient method (GRG) for solving NLPs involving nonlinear constraints. See [10.2, 10.13, 10.15, 10.17, 10.25, 10.26].

10.10.5 The Gradient Projection Method

When applied to solve the NLP (10.37), this method generates a descent sequence $\{x^r : r = 0, 1, \ldots\}$ beginning with a feasible point x^0, all the points in which are feasible. Step 1 begins with x^0, and in general for $k \geqq 1$ step $k+1$ begins with the point x^k at the end of step k.

For any feasible solution $\overline{x}$ of (10.37) define $\mathbf{I}(\overline{x}) = \{i : A_{i.}\overline{x} = b_i\}$. Clearly, $\{1, \ldots, m\} \subset \mathbf{I}(\overline{x})$ for all feasible solutions $\overline{x}$.

In step $k+1$, if there are no equality constraints in the problem and if $\mathbf{I}(x^k) = \emptyset$, choose the search direction at x^k to be $y^k = -(\nabla\theta(x^k))^T$. If $\mathbf{I}(x^k) \neq \emptyset$, the search direction in this step is determined by projecting the negative gradient of the objective function at x^k, onto the subspace parallel to the affine space of currently active constraints treated as equations. Let A_k denote the matrix whose rows are $A_{i.}$ for $i \in \mathbf{I}(x^k)$. So A_k is of order $|\mathbf{I}(x^k)| \times n$. Assume that the set of rows of A_k is linearly independent, otherwise delete some dependent row vectors of A_k from it until this property holds. The projection matrix corresponding to the active subspace is $P_k = I - A_k^T(A_kA_k^T)^{-1}A_k$. The projection of $-(\nabla\theta(x^k))^T$ onto the active subspace is $-P_k(\nabla\theta(x^k))^T$. It can be verified that this vector $-P_k(\nabla\theta(x^k))^T$ is a positive multiple of the vector which minimizes $(\nabla\theta(x^k))y$ subject to $A_ky = 0$ and $y^Ty \leqq 1$.

If $-P_k(\nabla\theta(x^k))^T = 0$, define $\beta^k = (A_kA_k^T)^{-1}A_k(\nabla\theta(x^k))^T$. Then $\nabla\theta(x^k) - (\beta^k)^TA_k = 0$. $(\beta^k)^T$ is a row vector of dimension $|\mathbf{I}(x^k)|$. Augment $(\beta^k)^T$ into a vector of dimension $m+p$, by inserting 0's for all $i \notin \mathbf{I}(x^k)$, and let the vector obtained be called π^k. Then $\nabla\theta(x^k) = \pi^kA$ where A is the $(m+p)\times n$ coefficient matrix in (10.37). So if $\pi_i^k \geqq 0$ for all $i = m+1$ to $m+p$, x^k, π^k together satisfy the first order necessary optimality conditions (10.38) and the method terminates with x^k as the KKT point for (10.37). On the other hand if $\pi_i^k < 0$ for some i between $m+1$ to $m+p$, identify the i for which π_i^k is the most negative, say r, delete the rth constraint from the active set (that is, eliminate $A_{r.}$ from the matrix A_k) update the projection matrix, and the projection of $-(\nabla\theta(x^k))^T$ on the new active subspace, and repeat the whole process.

If $-P_k(\nabla\theta(x^k))^T \neq 0$, define $y^k = -P_k(\nabla\theta(x^k))^T$, y^k is the search direction at x^k. It can be verified that P_k is symmetric and $P_k^T P_k = P_k$, so P_k is PSD. Also $\nabla\theta(x^k)y^k = -\|y^k\|^2 < 0$. So y^k is a descent direction. Now find $\overline{\lambda}$ from

$$\begin{aligned}\overline{\lambda} = &\text{ minimum } \left\{\frac{A_{i.}x^k - b_i}{-A_{i.}y^k} : i \text{ such that } i \not\in \mathbf{I}(x^k) \text{ and } A_{i.}y^k < 0\right\}\\ &= +\infty \text{ if } A_{i.}y^k \geqq 0 \text{ for all } i \not\in \mathbf{I}(x^k).\end{aligned}$$

Do a line search to minimize $\theta(x^k + \lambda y^k)$, $0 \leqq \lambda \leqq \overline{\lambda}$. If λ_k is the optimum step length in this line search problem, $x^{k+1} = x^k + \lambda_k y^k$ is the new point; go to the next step.

Methods for Updating the Projection Matrices

The periodic updating of the projection matrix is a considerable computational problem. However, the matrix A_k usually changes by one row, say $A_{r.}$, which is either dropped from the set of active constraint rows, or is added to it. Here we discuss how to efficiently update $(A_k A_k^T)^{-1}$ when a change like this takes place.

To Delete a Row From A_k

Let $A_{r.}$ be the sth row in A_k at the moment and suppose we want to delete it from A_k. After deletion suppose A_k becomes $\hat{A}$, of order $(q-1) \times n$.

Interchange the last row and the sth row in $(A_k A_k^T)^{-1}$. In the resulting matrix interchange the sth column and the last column. After these interchanges suppose this matrix $(A_k A_k^T)^{-1}$ is written down in partitioned form as

$$\begin{pmatrix} E & u \\ u^T & \delta \end{pmatrix}$$

where E is of order $(q-1) \times (q-1)$. Then it can be shown that $(\hat{A}\hat{A}^T)^{-1} = E - \frac{uu^T}{\delta}$.

To Add a Row to A_k

Suppose the row $A_{r.}$ has to be added to A_k. We will make $A_{r.}$ as the last row of the resulting matrix, which is $\tilde{A} = \begin{pmatrix} A_k \\ A_{r.} \end{pmatrix}$. Let $\overline{P}$ be the projection matrix corresponding to A_k, which is $I - A_k^T(A_k A_k^T)^{-1}A_k$. Compute $c = \|\overline{P}(A_{r.})^T\|^2 = A_{r.}\overline{P}(A_{r.})^T$, $w = (A_k A_k^T)^{-1}A_k(A_{r.})^T$, $u = -(w/c)$, $F = (A_k A_k^T)^{-1} + \frac{ww^T}{c}$. Then

$$(\tilde{A}\tilde{A}^T)^{-1} = \begin{pmatrix} F & u \\ u^T & 1/c \end{pmatrix}.$$

In the process of this updating, if c turns out to be zero, i. e., $\overline{P}A_{r.} = 0$, then the new active constraint row, $A_{r.}$, is linearly dependent on the previous active constraint rows, and the updating cannot be carried out. In this case the new active constraint row is ignored and the method can be continued with the same set of active constraint rows as before.

The updating procedure can also be used recursively to obtain the inverse $(A_k A_k^T)^{-1}$ in the first step of the algorithm, from the set of active constraints at that stage, by introducing them one at a time. An advantage of this recursion is that it selects the largest set of linearly independent active constraint rows among the set of all active constraint rows at this stage.

10.10.6 The Active Set Methods

We consider the NLP (10.37). These methods begin with a feasible solution x^0 and obtain a descent sequence $\{x^r : r = 0, 1, \dots\}$, where each point in the sequence is feasible.

If $\overline{x}$ is an optimum solution for (10.37), and $\mathbf{I}(\overline{x}) = \{i : A_{i.}\overline{x} = b_i, i = 1 \text{ to } m+p\}$, then $\overline{x}$ is also an optimum solution of the equality constrained NLP

$$\begin{array}{ll} \text{minimize} & \theta(x) \\ \text{subject to} & A_{i.}x = b_i, i \in \mathbf{I}(\overline{x}). \end{array} \tag{10.48}$$

If we can guess the correct active set $\mathbf{I}(\overline{x})$, we could solve (10.48) by methods for solving equality constrained NLPs discussed in Section 10.9.

In these methods, a guess is built up over the steps, on the likely set of active constraint indices at the optimum. This set is known as the **working active set**. The working active set in step $k+1$ is denoted by $\mathbf{I}_k$. Clearly $\{1, \dots, m\} \subset \mathbf{I}_k$ for all k. Changes are made in the set $\mathbf{I}_k$ using information gathered in each step. $\mathbf{I}_k$ always satisfies the property: $\{A_{i.} : i \in \mathbf{I}_k\}$ is linearly independent. The initial point in step 1 is x^0, in initial feasible solution with which the method is initiated. For $k \geqq 1$, the initial point in step $k+1$ is x^k, the feasible point obtained at the end of step k. Usually we have $\mathbf{I}_k \subset \mathbf{I}(x^k)$.

In step $k+1$, we carry a step for the equality constrained minimization problem

$$\begin{array}{ll} \text{minimize} & \theta(x) \\ \text{subject to} & A_{i.}x = b_i, i \in \mathbf{I}_k \end{array} \tag{10.49}$$

as discussed in Section 10.9. The search direction at x^k is the direction determined using the projected gradient, the projected Hessian or some quasi-Newton search direction at x^k for (10.49) as discussed in Section 10.9.

If x^k satisfies the termination criteria for (10.49), let A_k denote the matrix with rows $A_{i.}$, $i \in \mathbf{I}_k$. The corresponding Lagrange multiplier vector for (10.49) is $\beta^k = (A_k A_k^T)^{-1} A_k (\nabla\theta(x^k))^T$ from (10.31). If $\beta_i^k \geqq 0$ for all $i \in \mathbf{I}_k \cap \{m+1, \dots, m+p\}$, as discussed in Section 10.9.1, x^k is a KKT point for (10.37), terminate. If $\beta_i^k < 0$ for some $i \in \mathbf{I}_k \cap \{m+1, \ldots, m+p\}$, select the most negative among these, say β_r^k, then delete r from the working active set, and repeat the whole process.

If x^k does not satisfy the termination criteria for (10.49), let y^k be the search direction generated at x^k for solving (10.49). Find out $\overline{\lambda}$ from

$$\begin{aligned}\overline{\lambda} &= \text{ minimum } \left\{\frac{A_{i.}x^k - b_i}{-A_{i.}y^k} : i \text{ such that } i \notin \mathbf{I}_k \text{ and } A_{i.}y^k < 0\right\} \\ &= \infty \text{ if } A_{i.}y^k \geqq 0 \text{ for all } i \notin \mathbf{I}_k.\end{aligned}$$

Do a line search to minimize $\theta(x^k + \lambda y^k)$ over $0 \leqq \lambda \leqq \overline{\lambda}$. Let λ_k be the optimum step length for this line search problem. If $\lambda_k < \overline{\lambda}$, leave the working active set $\mathbf{I}_k$ unchanged, and with $x^{k+1} = x^k + \lambda_k y^k$ go to the next step. If $\lambda_k = \overline{\lambda}$, all the i which tie for the minimum in the definition of $\overline{\lambda}$ join the active set, select one of these and include it in $\mathbf{I}_k$. Then go to the next step.

To carry out a step of the algorithm discussed in Section 10.9 for the equality constrained minimization problem (10.49), we need the corresponding matrix Z, which we denote by Z_k here, as discussed in Section 10.9. Whenever we change the working active set $\mathbf{I}_k$ by dropping an element from it, or including a new element in it, it is necessary to make the corresponding changes in Z_k. Suppose Z_k is computed as in (10.27) using a basis B_k for A_k, and maintained by storing B_k either explicitly or in some factored form. Whenever $\mathbf{I}_k$ changes by one element, B_k changes by one row and one column, and B_k^{-1} can be updated by using the standard pivot methods of LP.

Several practical strategies have been developed to decide when to include a constraint in the working active set, and when to drop a constraint from it. Software packages for linearly constrained nonlinear programming based on such active set strategies seem to give the most satisfactory performance. Many of the commercially available packages usually include a combination of several of the strategies discussed above, in order to satisfactorily solve the widest class of problems.

All these methods become considerably simplified when applied to solve a quadratic programming problem, because of the special nature of the objective function.

10.11 Exercises

10.2 Fermat's Problem

Let $A_{.j} = (a_{1j}, \ldots, a_{mj})^T$, $j = 1$ to n be given distinct points in $\mathbf{R}^m$. Let w_j be a given positive weight associated with point $A_{.j}$. For any $x \in \mathbf{R}^m$ define $f(x) = \sum_{j=1}^n w_j \|x - A_{.j}\|$.

(i) If no three points among $\{A_{.1}, \ldots, A_{.n}\}$ are collinear, prove that $f(x)$ is positive and strictly convex on $\mathbf{R}^m$.

(ii) Assuming that no three points in the set $\{A_{.j} : j = 1 \text{ to } n\}$ are collinear prove that the problem of minimizing $f(x)$ over $\mathbf{R}^n$ has a unique solution, call it $\overline{x}$, and prove that $\overline{x}$ lies in the convex hull of $\{A_{.1}, \dots, A_{.n}\}$.

(iii) Define

$$g(x) = \sum_{j=1}^{n} \Big(\frac{w_j (A_{.j} - x)}{\|x - A_{.j}\|} \Big), \quad \text{if } x \neq A_{.j}, \text{ for each } j = 1 \text{ to } n.$$

For such points, $g(x) = -\nabla f(x)$. This function $g(x)$ given above, is not defined if $x = A_{.j}$ for some j. By analogy, define for $j = 1$ to n,

$$\begin{aligned} h(A_{.j}) &= \sum_{\substack{i=1 \\ i \neq j}}^{n} \Big(\frac{w_i (A_{.i} - A_{.j})}{\|A_{.i} - A_{.j}\|} \Big) \\ g(A_{.j}) &= \text{ maximum } \{\|h(A_{.j})\| - w_j, 0\} \Big(\frac{h(A_{.j})}{\|h(A_{.j})\|} \Big). \end{aligned}$$

Prove that a given point x is $\overline{x}$ (whether x is one of the points in the set $\{A_{.j} : j = 1 \text{ to } n\}$ or not) iff $g(x) = 0$, with $g(x)$ defined as above.

(iv) Assume that no three points in the set $\{A_{.j} : j = 1 \text{ to } n\}$ are collinear.
Define:

$$\begin{aligned} T(x) &= \Big(\sum_{j=1}^{n} \frac{w_j A_{.j}}{\|x - A_{.j}\|} \Big) / \Big(\sum_{j=1}^{n} \frac{w_j}{\|x - A_{.j}\|} \Big), \quad \text{if } x \neq A_{.j} \text{ for each } j = 1 \text{ to } n \\ T(A_{.j}) &= A_{.j}, \text{ for each } j = 1 \text{ to } n. \end{aligned}$$

Prove that $T(\overline{x}) = \overline{x}$. Also prove that if x is such that $x \neq A_{.j}$ for each $j = 1$ to n and $T(x) = x$, then $x = \overline{x}$.
Prove that if $x \in \mathbf{R}^m$ satisfies $x \neq T(x)$, then $f(T(x)) < f(x)$.
Consider the interative method $x^0 =$ initial point in $\mathbf{R}^m$ choosen so that

$$\begin{aligned} x^0 &\neq A_{.j} \text{ for each } j = 1 \text{ to } n \\ x^{r+1} &= T(x^r), r = 0, 1, \dots. \end{aligned}$$

If $x^r \notin \{A_{.j} : j = 1 \text{ to } n\}$ for all r, prove that the sequence $\{x^r : r = 0, 1, \dots\}$ converges to $\overline{x}$.

(v) Let $A_{.j}$ be the jth column vector of the following matrix for $j = 1$ to 6.

$$\begin{pmatrix} -2 & -1 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}.$$

Let $w_j = 1$ for all $j = 1$ to 6. In this case prove that $\overline{x} = (0, 0)^T$.
Show that there is an x_1^0 (approximately 1.62) such that for $x^0 = (x_1^0, 0)^T$ we have $T(x^0) = A_{.3}$, which is not optimal. This shows that the iterative method discussed in (iv) may not always converge to $\overline{x}$ even if the initial point $x^0 \notin \{A_{.j} : j = 1 \text{ to } n\}$.

However, prove that there exists a countable set $\mathbf{\Gamma}$ of points in $\mathbf{R}^m$ such that if $x^0 \notin \mathbf{\Gamma}$, then the sequence of points generated by the iterative method discussed in (iv) converges to $\overline{x}$.
(H. W. Kuhn [10.21])

10.3 Consider the NLP

$$\begin{array}{ll} \text{minimize} & \theta(x) \\ \text{subject to} & f(x) = 0 \end{array}$$

where $\theta(x)$ and $f(x)$ are both continuously differentiable real valued functions defined over $\mathbf{R}^n$. Using the ideas of the reduced gradient method and the results given by the implicit function theorem, develop an efficient algorithm for solving this problem.

10.4 Define the diameter of a convex hexagon (convex polytope with six extreme points in $\mathbf{R}^2$) $\mathbf{K}$ to be maximum $\{\|x - z : x, z \in \mathbf{K}\}$. Formulate the problem of finding a maximum area convex hexagon of diameter $\leqq 1$, as an NLP. Is this a convex programming problem? Find a solution to this problem using some of the algorithms discussed in this book.

10.5 $D = (d_{ij})$ is a square symmetric matrix of order n satisfying, $d_{ii} = 0$ for all i, the triangle inequality ($d_{ij} + d_{jk} \geqq d_{ik}$ for all i, j, k), and $d_{ij} > 0$ for all $i \neq j$. It is the matrix of Euclidean distances between pairs of points among a set of n points in $\mathbf{R}^2$. We are given the matrix D, but not the actual points from which D was calculated. It is required to find the coordinates (x_i, y_i), $i = 1$ to n, of n points in $\mathbf{R}^2$, for which the pairwise distance matrix is D. Formulate this as an NLP and discuss an efficient approach for solving it.

The rectilinear or L_1-distance between two points (x_1, y_1), (x_2, y_2) in $\mathbf{R}^2$ is defined to be $|x_1 - x_2| + |y_1 - y_2|$. Consider the version of the above problem of finding the coordinates of n points in $\mathbf{R}^2$, for which the matrix of pairwise rectilinear distances is a given matrix D. Formulate this problem. Is this easier or harder to solve than the version for the Euclidean distances? Why?
(S. M. Pollock)

10.6 Let $n > 1$, $x = (x_1, \ldots, x_n)^T$, $S_k = \sum_{j=1}^{n} x_j^{\mathbf{k}}$. Consider the NLP

$$\begin{array}{ll} \text{minimize} & S_3^{\mathbf{2}} - S_2 S_4 \\ \text{subject to} & 0 \leqq x_j \leqq 1,\ j = 1 \text{ to } n. \end{array}$$

Prove that the vector $x = (x_j)$ is a strict local minimum for this problem if m of the x_j are equal to 1, and p of the x_j are equal to 1/2, where $m + p = n$ and $n > m > (1/9)n$. Also, prove that x is a global minimum for this problem if it is of the above form and either m or p is $\lfloor (1/2)n \rfloor$.
(P. Wolfe [10.41])

10.7 Automatic Voltage Regulator Control Panel (AVR) Design Problem. AVR's are used to stabilize voltage in electrical power systems. AVR contains many circuits, each circuit may consist of several components like resistors, transistors, capacitors, zener diodes etc. Each component is characterized by some variables (e. g. the resistence of a resistor measured in ohms, the gain value of a transistor measured in hFE, the capacitance of a capacitor measured in microfared (MF) etc.). The problem is to find an optimum design (i. e., find the optimal values of all the variables) which stabilizes the output voltage as far as possible, while the input voltage may fluctuate uncontrollably in some specified range. Here we provide a simplified example relating to the triggering circuit design in the AVR control panel for a diesel 2MW AC generator, to illustrate the basic principles involved in modelling and solving this class of problems (the general problem may have many more variables, and the functions involved are more complicated and may have many more terms, but the basic features remain identical). The functional form for the output voltage as a function of the input voltage and the design variables is available from electrical engineering theory. Given this function, and the range of fluctuation of the input voltage, the problem is to find optimal values for the design variables that stabilizes the output voltage as much as possible. In our example, the positive and negative voltages are denoted by v_1, v_2; each of these fluctuates between 14.25 to 15.75 and we have no way of controlling it. There are 5 design variables, x_1, x_2, x_3, x_4, x_5. The functional form for the output voltage v is the following:

$$\begin{aligned} v_3 &= v_1(1 - \mathrm{e}^{-(\mathbf{0.5/x_1x_5})}) \\ v_4 &= (x_4(x_3 + 100) + 100v_2)/(x_3 + 200) \\ v &= (v_3 - v_4)\mathrm{e}^{-(\mathbf{10/x_2x_5})}. \end{aligned}$$

The constraints on the variables are, $1 \leqq x_5 \leqq 10$, $3 \leqq x_4 \leqq 15$, $10 \leqq x_1 \leqq 200$, $100 \leqq x_2 \leqq 4000$, $1 \leqq x_3 \leqq 1000$. Formulate the problem as a nonlinear program and discuss an algorithm for solving it.
(Kirloskar Electricals Ltd., India)

10.8 The variable y represents the yield in a chemical process. There are n process variables $x_1, x_2, \dots, x_n$ (such as temperature, flow rate, etc.) which influence the yield y. Data was collected to observe the yield y for various values of the process variable vector $x = (x_1, \dots, x_n)$. This leads to k data points, $t = 1$ to k.

Process variable vector $x^t = (x_1^t, \dots, x_n^t)$, corresponding yield y_t.

In the vectors x^t, $t = 1$ to k, each process variable takes several values spanning its possible range of variation, and each combination of process variables takes several values in the combined range of variation of the vector of these process variables. It is believed that y can be approximated reasonably well by a convex quadratic function of the form $Q(x) = cx + (\frac{1}{2})x^T Dx$. It is required to find the best convex quadratic fit $Q(x)$ for y, using the available data. Formulate this problem of finding the best convex quadratic approximation $Q(x)$ for y using the available data as a nonlinear programming problem, and discuss how this problem can be solved.

If l_j, u_j are known lower and upper bounds for the process variable x_j for $j = 1$ to n, and you are asked to design an experiment for collecting the necessary data in the above problem, outline how you will determine the process variable vectors $x^t = (x^t_1, \ldots, x^t_n)$ at which the yield has to be observed, in order to obtain the best fit.

10.9 Let $\theta(x)$ be a continuously differentiable real valued function defined on $\mathbf{R}^n$. It is required to find the unconstrained minimum of $\theta(x)$ over $\mathbf{R}^n$. Beginning with an initial point $x^0 \in \mathbf{R}^n$, the sequence of points $\{x^r : r = 0, 1, 2, \ldots\}$ was obtained by using Cauchy's Method of steepest descent with optimal step lengths in each step (the metric matrix for determining the steepest descent direction is always the unit matrix I). Prove that $(x^{r+2} - x^{r+1})^T(x^{r+1} - x^r) = 0$ for all r.

10.10 Let c be a given row vector in $\mathbf{R}^n$. Write down explicitly, an optimum solution for the following problem

$$\begin{array}{ll} \text{minimize} & cx \\ \text{subject to} & x^T x = 1 \\ & \quad x \geqq 0. \end{array}$$

10.11 Let $\theta(x)$ be a continuously differentiable real valued convex function defined on a bounded convex set $\mathbf{K} \subset \mathbf{R}^n$, that attains its minimum over $\mathbf{K}$ at $x^* \in \mathbf{K}$. $\{x^r : r = 1, 2, \ldots\}$, $\{y^r : r = 1, 2, \ldots\}$ are sequences of points in $\mathbf{K}$ satisfying the following conditions

$$\nabla\theta(x^r)(y^r - x^r) \leqq \text{ Infimum } \{\varepsilon_r + \nabla\theta(x^r)(x - x^r) : x \in \mathbf{K}\}$$

$$\nabla\theta(x^r)(y^r - x^r) \to 0 \text{ as } r \to \infty$$

where $\varepsilon_r \geqq 0$ for all r and $\varepsilon_r \to 0$ as $r \to \infty$. Then, prove that $\theta(x^r) \to \theta(x^*)$ as $r \to \infty$.

10.12 We are given a set of n points in $\mathbf{R}^2$, say, $a^t = (a^t_1, a^t_2)$, $t = 1$ to n. It is required to fit a circle to these points. The objective function to be minimized is $\sum$ (($r^\mathbf{2}$-square of the Euclidean distance between a^t and the center)$^\mathbf{2}$: $t = 1$ to n), where r is the radius of the circle. Formulate this problem as an NLP and discuss an efficient method for solving it.
(R. Chandrasekaran)

10.13 We are given row vectors $c^1, \ldots, c^r$ in $\mathbf{R}^n$ and real numbers $d_1, \ldots, d_r$. Define

$$\theta(x) = \text{ Maximum } \{|c^t x - d_t| : t = 1 \text{ to } r\}.$$

It is required to find the unconstrained minimum of $\theta(x)$ over $x \in \mathbf{R}^n$. Discuss an efficient method for computing it.

10.14 Let $d_1, \ldots, d_n$ be given positive integers. The partition problem with this data, is to check whether there exists a subset $\mathbf{S} \subset \{1, \ldots, n\}$ such that

$$\sum_{i \in \mathbf{S}} d_i = \sum_{i \in \overline{\mathbf{S}}} d_i$$

where $\overline{\mathbf{S}} = \{1, \ldots, n\} \setminus \mathbf{S}$. This is a well known $\mathcal{NP}$-complete problem (see [8.12]). Formulate this problem as a special case of

$$\begin{array}{ll} \text{minimize} & \|x\|_p \\ \text{subject to} & x \in \mathbf{K} = \{x : Ax \geqq b\} \end{array} \tag{10.50}$$

where $\|x\|_p = (\sum_{i=1}^n |x_i|^{\mathbf{p}})^{\mathbf{1/p}}$, and A, b are integer matrices of orders $m \times n$ and $m \times 1$ respectively, p a positive integer $\geqq 1$, and $\mathbf{K}$ is known to be nonempty and bounded. $\|x\|_p$ is known as the p-norm of the vector x. Thereby establish that the problem of maximizing the p-norm on a convex polytope specified in terms of linear inequalities with integer data, is an $\mathcal{NP}$-hard problem.

Show that an upper bound on the optimum objective value in (10.50) can be obtained by solving a relaxed linear program.

The ∞-norm of the vector $x = (x_i) \in \mathbf{R}^n$, denoted by $\|x\|_\infty$ is defined to be maximum $\{|x_i| : i = 1 \text{ to } n\}$. Show that when $p = \infty$, (10.50) can be solved by solving at most $2n$ linear programs.
(O. L. Mangasarian and T.-H. Shiau, "A variable-complexity norm maximization problem", Technical Report 2780, Mathematics Research Center, University of Wisconsin, Madison, 1984)

10.15 Optimal Betting in a Race Track
The "market" at a race track in North America typically convenes for about 20 minutes, during which participants make bets on any number of 6 to 12 horses in the following race. To keep the discussion simple, we consider a race in which participants can bet on each horse either to **win** or **place**. All participants who have bet on a horse to **win**, realize a positive return on that bet only if that horse comes first, while a **place** bet realizes a positive return if that horse comes first or second. Consider a race with the following data declared at the time we are ready to bet.

n	=	number of horses running in the race.
W_i	=	total amount bet by public (all participants so far) on horse i to **win**.
W	=	$\sum_{i=1}^n W_i$ = **win** pool.
Q	=	track payback proportion (typically .83, it is the proportion of pool given away; the remaining proportion .17 is kept by the race track company).
P_j	=	total amount bet by public (all participants so far) on horse j to **place**.
P	=	$\sum_{j=1}^n P_j$ = **place** pool.

q_i = probability that horse i finishes first in the race.

q_{ij} = $\frac{q_i q_j}{1-q_i}$ = probability that horse i finishes first and horse j finishes second in the race.

$$\begin{array}{l}\text{payoff per dollar bet}\\ \text{on horse } i \text{ to } \textbf{win}\end{array} = \begin{cases} \frac{WQ}{W_i}, & \text{iff horse } i \text{ comes in first place} \\ 0 & \text{otherwise.}\end{cases}$$

$$\begin{array}{l}\text{payoff per dollar bet}\\ \text{on horse } j \text{ to } \textbf{place}\end{array} = \begin{cases} 1 + \frac{PQ-P_i-P_j}{2P_j}, & \text{if horses } i, j \text{ are first two winners in any order} \\ 0, & \text{if horse } j \text{ did not finish in the first two places in the race.}\end{cases}$$

Thus the payoff on horse j to **place** is independent of whether j finishes first or second, but dependent on which horse finishes with it in first two places.

We assume that $q_i = W_i/W$, that is that the crowd is good at picking a winner, or that the relative amount bet on a horse to **win** corresponds closely to its actual chances of winning.

The W_i, P_i are the public's bets in the race, are known. Consider the problem of determining the place bets to make optimally, given all the above data and the assumptions, subject to a budget of b \$. The Kelly criterion determines the optimal bets to maximize the expected logarithm of final wealth. The decision vector in this problem is $x = (x_1, \ldots, x_n)^T$, where x_i is the **place** bet on the ith horse, $i = 1$ to n. Define

$$f_{ij}(x) = \Big(\frac{Q\big(P + \sum_{l=1}^{n} x_l\big) - x_i - x_j - P_i - P_j}{2}\Big)\Big(\frac{x_i}{x_i + P_i} + \frac{x_j}{x_j + P_j}\Big).$$

Then the problem for determining the optimal x is

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} q_{ij}\log\Big(f_{ij}(x) + b - \sum_{\substack{l=1\\ l\neq i,j}}^{n} x_l\Big) \\ \text{subject to} & \displaystyle\sum_{l=1}^{n} x_l \leqq b \\ & \quad x_l \geqq 0, \ \text{for all } l = 1 \text{ to } n. \end{array}$$

Discuss an efficient approach for solving this problem. Solve the numerical problem using this approach, when the data is

$$n = 8, \ Q = 0.83, \ b = 500.$$

i	1	2	3	4	5	6	7	8
W_i	10,000	15,000	5,000	35,000	5,000	10,000	18,000	12,000
P_i	4,000	4,000	4,000	8,000	3,000	8,000	13,000	5,000

(See the delightful book, W. T. Ziemba and D. B. Hausch [10.42] for a complete treatment of this problem.)

10.16 Consider the following LP

$$\begin{array}{ll} \text{minimize} & cx \\ \text{subject to} & x \in \mathbf{K} = \{x : Ax \geqq b\} \end{array}$$

where A, b are given matrices of orders $m \times n$, $m \times 1$ respectively. Assume that $\mathbf{K} \neq \emptyset$. For $\lambda \geqq 0$, let $x(\lambda)$ denote the nearest point in $\mathbf{K}$ to λc in terms of the usual Euclidean distance.

If the above LP has an optimum solution, prove that there exists a $\overline{\lambda} > 0$ such that $x(\lambda)$ is constant for all $\lambda \geqq \overline{\lambda}$ and that $x(\overline{\lambda})$ is the least (Euclidean) norm optimum solution for the LP.

If the objective value is unbounded below on $\mathbf{K}$ in the above LP, prove that $\|x(\lambda)\| \to \infty$ as $\lambda \to \infty$.
(O. L. Mangasarian)

10.17 Consider the following NLP

$$\begin{array}{ll} \text{minimize} & \theta(x) \\ \text{subject to} & Ax \geqq b \end{array}$$

where $\theta(x)$ is a strictly convex function defined over $\mathbf{R}^n$ with a unique unconstrained minimum, $\overline{x}$, in $\mathbf{R}^n$; and A is a matrix of order $m \times n$. Suppose $\overline{x}$ satisfies

$$A_{i.}x - b_i \begin{cases} < 0, & i = 1 \text{ to } r \\ \geqq 0, & \text{for } i = r+1 \text{ to } m. \end{cases}$$

Let x^i denote the point which minimizes $\theta(x)$ subject to one constraint only "$A_{i.}x \geqq b_i$", for $i = 1$ to r. Suppose there is a unique $k \in \{1, \ldots, r\}$ such that x^k is feasible to the original NLP. Then prove that x^k is an optimum solution for the original NLP.

10.18 A Curve Fitting Application in High Voltage Coil Insulation Testing: The life of the insulation system on high voltage coils used in rotating electrical machines, depends on it's DLA (dielectric loss analyzer) value. The DLA value for a coil is expected to depend on it's $\Delta \tan \delta$ (increase in $\tan \delta$ or dissipation factor expressed as a percentage, with increase in test voltage) and ΔC (inrease in capacitance with increase in test voltage) values. The DLA value is hard to measure, but the $\Delta \tan \delta$ and ΔC values can be measured easily. Given below are the DLA, $\Delta \tan \delta$ and ΔC values for a sample of 95 test coils. Use this data to determine if the DLA value of a coil can be estimated reliably from it's $\Delta \tan \delta$ and ΔC values, and if so, determine the appropriate functional form. Using this analysis, design a scheme for checking the acceptability of coils (acceptable if DLA value is $\leqq 8.0$ units) using measurements of their $\Delta \tan \delta$ and ΔC values as far as possible.

ΔC and $\Delta \tan\delta$ with their corresponding DLA values for 95 test coils.

Sample coil No.	$\Delta \tan\delta$, 6.6 KV to 11 KV	Δ C, 6.6 KV to 11 KV	DLA at 11 KV	Sample coil No.	$\Delta \tan\delta$, 6.6 KV to 11 KV	Δ C, 6.6 KV to 11 KV	DLA at 11 KV
1	.0011	0.5	0.4	26	.0044	2.4	2.5
2	.0017	0.9	0.8	27	.0042	2.5	1.6
3	.0030	1.0	1.6	28	.0041	2.5	2.2
4	.0019	1.2	0.8	29	.0048	2.6	2.0
5	.0020	1.3	0.2	30	.0042	2.7	1.5
6	.0026	1.3	1.1	31	.0060	2.7	1.7
7	.0020	1.4	1.2	32	.0039	2.7	1.4
8	.0028	1.6	1.4	33	.0030	2.7	1.2
9	.0023	1.6	1.4	34	.0031	2.8	2.3
10	.0027	1.7	1.6	35	.0047	2.9	3.0
11	.0024	1.7	1.6	36	.0052	3.3	2.7
12	.0023	1.8	1.0	37	.0036	3.3	2.1
13	.0032	1.9	2.1	38	.0049	3.3	2.4
14	.0026	1.9	1.5	39	.0045	3.3	2.5
15	.0027	2.0	1.6	40	.0053	3.3	2.4
16	.0026	2.0	1.2	41	.0050	3.6	3.7
17	.0031	2.0	2.8	42	.0054	3.6	2.9
18	.0041	2.0	0.6	43	.0056	3.7	4.0
19	.0045	2.1	0.6	44	.0059	3.8	2.4
20	.0032	2.1	2.1	45	.0057	3.8	2.5
21	.0031	2.1	1.2	46	.0057	3.9	3.5
22	.0024	2.2	1.5	47	.0067	4.1	3.3
23	.0031	2.2	1.4	48	.0045	4.3	3.5
24	.0028	2.2	1.4	49	.0059	4.5	4.0
25	.0029	2.4	1.1	50	.0066	4.5	3.4

Sample coil No.	$\Delta \tan\delta$, 6.6 KV to 11 KV	Δ C, 6.6 KV to 11 KV	DLA at 11 KV	Sample coil No.	$\Delta \tan\delta$, 6.6 KV to 11 KV	Δ C, 6.6 KV to 11 KV	DLA at 11 KV
51	.0076	4.5	3.4	71	.0045	6.2	5.1
52	.0073	4.6	4.2	72	.0066	6.2	5.2
53	.0058	4.6	4.0	73	.0077	6.3	3.6
54	.0060	4.7	3.4	74	.0093	6.4	7.0
55	.0072	4.8	4.2	75	.0089	6.3	5.8
56	.0057	4.9	4.1	76	.0055	6.8	7.3
57	.0068	4.9	3.6	77	.0083	7.3	5.9
58	.0076	5.3	6.5	78	.0096	7.5	5.0
59	.0084	5.0	3.4	79	.0091	7.7	6.0
60	.0063	5.0	3.1	80	.0100	8.0	5.5
61	.0058	5.0	5.4	81	.0109	8.3	7.6
62	.0053	5.2	4.6	82	.0045	8.8	7.3
63	.0067	5.2	4.6	83	.0094	9.1	7.0
64	.0064	5.3	5.2	84	.0104	9.1	6.4
65	.0072	5.3	4.2	85	.0093	8.2	9.3
66	.0086	5.3	6.7	86	.0140	10.0	8.2
67	.0074	5.6	5.1	87	.0121	10.0	10.8
68	.0081	6.0	3.8	88	.0143	10.3	6.7
69	.0070	6.0	4.2	89	.0124	10.5	9.9
70	.0074	6.0	4.0	90	.0120	12.1	9.7
				91	.0131	11.1	10.4
				92	.0155	13.3	7.1
				93	.0127	14.6	9.0
				94	.0144	14.7	16.0
				95	.0139	15.4	13.2

10.12 References

10.1 L. Armijo, "Minimization of Functions Having Lipschitz Continuous First Partial Derivatives", *Pacific Journal of Mathematics*, 16 (1966) 1–3.

10.2 M. Avriel, *Nonlinear Programming, Analysis and Methods,* Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

10.3 M. S. Bazaraa and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms,* Wiley, New York, 1979.

10.4 J. Backen and G. P. McCormick, *Selected Applications of Nonlinear Programming,* Wiley, New York, 1968.

10.5 A. Ben-Israel, A. Ben-Tal and S. Zlobec, *Optimality in Nonlinear Programming: A Feasible Direction Approach,* Wiley-Interscience, New York, USA, 1981.

10.6 J. M. Borwein, "Necessary and sufficient conditions for quadratic minimality", *Numerical Function Analysis and Optimization*, 5 (1982) 127–140.

10.7 L. B. Contesse, "Une caratérisation complète des minima beaux en programmation Quadratique", *Numer. Math.,* 34 (1980) 315–332.

10.8 J. E. Dennis Jr., and J. J. Moŕe, "A Characterization of Superlinear Convergence and Its Application to Quasi-Newton Methods", *Mathematics of Computation*, 28 (1974) 549–560.

10.9 J. E. Dennis Jr., and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations,* Prentice-Hall, Englewood Cliffs, NJ, USA, 1983.

10.10 G. P. Egorychev, "Reshenie Problemy van der Wardena, dha Permanentov", Inst. Fiziki im. L. V. Kirenskogo, USSR Acad. Sci. Siberian branch, Preprint IFSO – 13M, Krasnoiarsk, 1980.

10.11 G. P. Egorychev, "Novye formuly dha Permanenta", *Dokl. Acad. Sci. USSR,* 254 (1980) 784–787.

10.12 A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques,* Wiley, New York, 1968.

10.13 R. Fletcher, *Practical Methods of Optimization, Vols. 1,2,* Wiley-Interscience, New York, 1981.

10.14 N. Frank and P. Wolfe, "An Algorithm for Quadratic Programming", *Naval Research Logistics Quarterly*, 3 (1956) 95–110.

10.15 P. E. Gill and W. Murray, *Numerical Methods for Cconstrained Optimization,* Academic Press, New York, 1974.

10.16 P. E. Gill, W. Murray, M. A. Sanders and M. H. Wright, "Model Building and Practical Implementations Aspects of Nonlinear Programming", Technical Report, Department of OR, Stanford University, Stanford, California, USA, 1984.

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

10.18 D. Goldfarb, "Extension of Davidson's Variable Metric Method to Maximization Under Linear Equality and Inequality Constraints", *SIAM Journal on Applied Mathematics*, 17 (1969) 739–764.

10.19 W. Hock and K. Schittkowski, "Test Examples for Nonlinear Programming Codes", *Lecture Notes in Economics and Mathematical Systems* Vol. 187, Springer-Verlag, New York (1981).

10.20 D. E. Knuth, "A Permanent Inequality", *The American Mathematical Monthly*, 88 (1981) 731–740.

10.21 H. W. Kuhn, "A Note on Fermat's Problem", *Mathematical Programming*, 4 (1973) 98–107.

10.22 C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

10.23 C. Lemarechal and R. Mifflin, "Global and Superlinear Convergence of an Algorithm for One-Dimensional Minimization of Convex Functions", *Mathematical Programming*, 24 (1982) 241–256.

10.24 M. E. Lenard, "A Computational Study of Active Set Strategies in Nonlinear Programming With Linear Constraints", *Mathematical Programming*, 16 (1979) 81–97.

10.25 F. A. Lootsma, *Numerical Methods for Nonlinear Optimization*, Academic Press, New York, 1972.

10.26 G. P. McCormick, *Nonlinear Programming, Theory, Algorithms and Applications*, Wiley-Interscience, 1983.

10.27 G. P. McCormick, "A Modification of Armijo's Step-Size Rule for Negative Curvature", *Mathematical Programming*, 13 (1977) 111-115.

10.28 R. Mifflin, "A Superlinearly Convergent Algorithm for One-Dimensional Constrained Minimization Problems With Convex Functions", *Mathematics of Operations Research*, 8 (1983) 185–195.

10.29 R. Mifflin, "Stationarity and Superlinear Convergence of an Algorithm for Univariate Locally Lipschitz Constrained Minimization", *Mathematical Programming*, 28 (1984) 50–71.

10.30 J. J. More' and D. C. Sorensen, "On the Use of Directions of Negative Curvature in a Modified Newton Method", *Mathematical Programming*, 16 (1979) 1–20.

10.31 B. A. Murtagh and M. A. Saunders, "Large-Scale Linearly Constrained Optimization", *Mathematical Programming*, 14 (1978) 41–72.

10.32 K. G. Murty and S. N. Kabadi, "Some $\mathcal{NP}$-complete Problems in Quadratic and Nonlinear Programming", *Mathematical Programming*, 38 (1987), to appear.

10.33 J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

10.34 P. Ribenboim, *13 Lectures on Fermat's Last Theorem,* Springer-Verlag, New York, 1980.

10.35 J. B. Rosen, "The gradient projection method for nonlinear programming, Part I: linear constraints", *SIAM J. Applied Math.,* 8 (1960) 181–217.

10.36 J. B. Rosen, "The gradient projection method for nonlinear programming, Part II: nonlinear constraints", *SIAM J. Applied Math.,* 9 (1961) 514–553.

10.37 L. E. Scales, *Introduction to Non-Linear Optimization,* Springer-Verlag, New York, 1985.

10.38 K. Schittkowski, *Nonlinear Programming Codes, Information, Tests, Performance,* Springer-Verlag, New York, 1980.

10.39 D. F. Shanno, "Conjugate Gradient Methods With Inexact Searches", *Mathematics of Operations Research*, 3 (1978) 244–254.

10.40 B. L. van der Waerden, "Problem, Jaresbericht", *Deutsch. Math. Verein*, 25 (1926) 117.

10.41 P. Wolfe, "Explicit Solution of an Optimization Problem", *Mathematical Programming*, 2 (1972) 258–260.

10.42 W. T. Ziemba and D. B. Hausch, *Beat the Race Track,* Harcourt Brace Jovanovich, New York, 1984.