Chapter 5

NONLINEAR PROGRAMMING - MULTIVARIABLE OPTIMIZATION PROCEDURES

Introduction

This part of optimization is the most dynamic topic. Applications are varied and appear in almost every field. Over the past three decades, the capability to locate local optima of a nonlinear economic model of a plant and to comply with several thousand constraints associated with the process models, unit capacities, raw material availabilities and product demands has been developed in proprietary codes of major corporations (1). Generally available nonlinear codes for large problems have grown from university and government research programs on numerical experimentation with algorithms, and high-level modeling language for mathematical programming and optimization, such as GAMS are now available. These modeling languages consist of a language compiler and a stable of integrated high-performance solvers tailored for complex, large scale modeling applications, and large maintainable models can be adapted quickly to new situations.

The capability to solve optimization problems with increasing numbers of constraints has grown with improvements in computer hardware and software. However, there still is debate about which algorithms and/or computer codes are superior; and Lasdon (3) has recommended having several codes available which implement some of the more successful methods.

The effectiveness of a multivariable optimization procedure depends on several, interrelated things. These are the optimization theory, the algorithms to implement the theory, the computer program and programming language used for computations with the algorithms, the computer to run the program, and the optimization problems being solved. For example, in the area of multivariable, unconstrained search methods; there are several hundred algorithms that have been used with varying degrees of success. They have been programmed in FORTRAN mainly, run on various types of computers and applied to a range of problems from simple algebraic expressions to plant simulation.

This chapter describes unconstrained and constrained multivariable search algorithms that have been successful in solving industrial optimization problems. Examples are given to illustrate these methods, and references to sources for computer programs are given for the methods. Also, references to recent and classical texts and articles are included for further information. For example, a two-volume set of books by Fletcher (4,5) is a recent comprehensive compilation of the mathematical aspects of nonlinear programming methods, as are the equally recent books by Gill, Murray and Wright (6), McCormick (7), and Bertsedkas (50). The books by Reklaitis, et. al. (15), Vanderplaat (24), Haftka and Kamat (54) and Dennis and Schnabel (55) describe the theory and recent computational practice, and Avriel's book (9) gives a broad mathematical coverage of the subject. Finally, Wilde's book (10), *Optimum Seeking Methods*, was the first book devoted to the subject, and it still contains valuable information in a very readable style.

In general form the nonlinear optimization problem can be stated as:

$$\text{Optimize:} \quad y(\mathbf{x}) \tag{5-1}$$

$$\text{Subject to:} \quad f_i(\mathbf{x}) = 0 \qquad \text{for } i = 1, 2, ..., h$$

$$f_i(\mathbf{x}) \geq 0 \qquad i = h+1, ..., m$$

There are n independent variables, $\mathbf{x} = (x_1, x_2, ... x_n)$, m constraint equations of which h are equality constraints. Also, the values of the x_j's can have upper and lower bounds specified. For this general form Avriel (11) points out that there is no unified approach to obtain the optimal solution of the nonlinear optimization problem that is comparable to the unifying role of the Simplex Method in linear programming. He states that the Simplex Method can efficiently solve a linear program in thousands of variables, but the question of how to minimize an unconstrained nonlinear function in more than a few variables is an important one.

There are *three classes of procedures* for multivariable optimization that are applicable to nonlinear economic models with nonlinear constraints. These are *multivariable search methods*, *multivariable elimination procedures*, and *stochastic methods*. Multivariable search methods have been the most important for process optimization and are discussed in detail. The capabilities and limitations of the other three methods are given in a summary form with reference to other sources for more complete information.

Multivariable search methods can be thought of as encompassing the theory and algorithms of nonlinear programming along with the associated computational methods. These procedures use algorithms that are based on geometric or logical concepts to move rapidly from a starting point away from the optimum to a point near the optimum. Also, they attempt to satisfy the constraints associated with the problem and the Kuhn-Tucker conditions, as they generate improved values of the economic model.

Multivariable elimination procedures are methods that reduce the feasible region (hypersurface of the independent variables) by discarding regions that are known *not* to contain the optimum (interval elimination). Some of these are similar to minimax single variable search methods in that they eliminate intervals on each of the independent variables. However, these methods are restricted to certain types of functions, e.g. strongly unimodal functions. Also, to locate the best value of the profit function with these procedures, the reduction in the range of the independent variables increases as the number of independent variables increases. This effect has been referred to as the *curse of dimensionality*, and it has been illustrated by Wilde (10). The single variable minimax interval elimination procedures are not useful in multi-dimensions since only line segments are eliminated in those procedures, and the number of lines in a plane is very large.

The more successful stochastic strategies include random search, genetic algorithms and simulated annealing. Random search is a stochastic method that places experiments randomly in

the feasible region after it has been divided into a grid of discrete points. Knowing the number and location of the grid points, a set of experiments is placed randomly. Then it can be determined with a certain probability that one of these points has a value of the profit function that is in a specified best fraction (top *x%*). Unimodality is not required, and the number of independent variables is not directly a factor. In adaptive or creeping random search, experiments are placed randomly in a selected section of the feasible region, and a best value is located. Then another section of the feasible region is placed around this best value, and random experiments are placed again. This procedure is repeated until a stopping criterion is met. In essence, random search is used as a multivariable search method.

Stochastic approximation procedures are methods that apply to economic models that contain random error, e.g. the plant instead of a computer simulation of the plant. These techniques are similar to multivariable search methods, but they move slowly to avoid being confounded by the random error in the values of the economic model.

These three methods are described such that they can be applied to industrial problems. The most important and widely used multivariable search methods are given first, and then the other three procedures are discussed.

Multivariable Search Methods Overview

Wilde (10) has proposed a strategy for multivariable search methods that contains some important ideas. This strategy has an opening gambit, a middle game and an end game that is analogous to the strategy of chess. In the opening gambit a starting point is selected. Then the middle game involves moving from this starting point to a point near the optimum as rapidly as possible. In the end game a quadratic fit to the economic model is performed to avoid stopping at a saddle point or sharp ridge.

Generally, selecting a starting point is not a problem for the current design or plant operating conditions are usually known. If they are not available, then midpoints between the upper and lower limits on the independent variables can be used, and Wilde (10) has suggested others such as the centroid and the minimax.

In the middle game a multivariable search method is used that moves rapidly from the starting point to a point that appears to be an optimum. Only enough local explorations are performed at each step to obtain information useful to locate future experiments and to keep the method moving rapidly toward the optimum. The objective is to attain a series of improved values of the economic model with a minimum of computational effort.

The end game takes over once the middle game procedure has located what appears to be an optimum. A quadratic fit to the economic model at this best point is performed to determine if it is an optimum rather than a saddle point or a ridge. The strategy has the middle game continue if an optimum is not located or stops if one is found based on the quadratic approximation.

With these ideas in mind, multivariable search methods will be described that are middle game procedures applicable to unconstrained and constrained problems. One of the more frequently encountered unconstrained optimization problems is that of a nonlinear least-squares fit of a curve to experimental data. However, industrial optimization problems are constrained ones, almost without exception. Moreover, it will be seen that some constrained methods convert the problem into an unconstrained one, and then an unconstrained procedure is employed. Also, some of the more effective middle game procedures develop the information for the quadratic fit of the end game as they proceed from the starting point.

There are several hundred unconstrained multivariable search methods, but most of them are variations on a few concepts. These concepts can be used to classify the methods. Many techniques may be called *geometric methods* for they use a local, geometric property to find a direction having an improved value of the economic model. Typically, derivative measurements are required. Two examples are the direction of steepest ascent (gradient search) and quadratic fit to the profit function (Newton's method). Other techniques can be called *logical methods* for they use an algorithm based on a logical concept to find an improved direction of the profit function. Two examples are pattern search and flexible polyhedron search. Typically, derivative measurements are not required; and these types of procedures also have been called *function comparison methods* (6). However, two methods that would not fit into these two categories readily are extensions of linear and quadratic programming. Here, linear programming, for example, is applied iteratively to a linearized version of the nonlinear constrained problem to move toward the optimum from a starting point. The methods are called *successive*, or sequential, *linear programming* and *successive*, or sequential, *quadratic programming*.

Another equally valid way to classify unconstrained methods has been given by Gill, Murray and Wright (6). These categories are Newton, quasi-Newton and conjugate gradient types, each with and without first or second derivatives, and functional comparison methods. Also, some of the quasi-Newton methods are called *variable metric* methods, and some of the conjugate gradient methods are called *conjugate direction* methods. They are all geometric methods, except for the functional comparison methods that are logical methods.

There are essentially six types of procedures to solve constrained nonlinear optimization problems. Four of these methods convert the constrained problem into an unconstrained one, and then an unconstrained search procedure is applied. These four types are penalty or barrier functions methods, the augmented Lagrange functions, generalized reduced gradients and feasible directions (or projections) sometimes called methods of restricted movement. The other two are the previously mentioned procedures of successive (or sequential) linear and quadratic programming.

Unconstrained Multivariable Search Methods

In this section on unconstrained multivariable search methods, several of the most effective and widely used methods are described. First, the quasi-Newton methods are given which have proved to be the most effective and more elaborate of the procedures. Then conjugate gradient and conjugate direction methods are illustrated with two examples. Finally, the popular function

comparison procedure, pattern search, is presented, and assessments of these methods are presented as related to problems with constraints.

Before discussing the specifics of the methods, it is necessary to describe the desirable features of an algorithm. As mentioned previously, the algorithm should generate a sequence of values of $\mathbf{x}_k$ that move rapidly from the starting point $\mathbf{x}_0$ to the neighborhood of the optimum $\mathbf{x}^*$. Then the iterates $\mathbf{x}_k$ should converge to $\mathbf{x}^*$ and terminate when a convergence test is satisfied. Therefore, an important theoretical result for an algorithm would be a theorem that proves the sequence of values $\mathbf{x}_k$ generated by the algorithm converges to a local optimum. For example, the following theorem from Walsh (26) provides sufficient conditions for convergence of the method of steepest ascent (gradient search).

If the limit of the sequence $\{\mathbf{x}_k\}$ *of* $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\nabla y(\mathbf{x}_k)$ *is* $\mathbf{x}^*$ *for all* $\mathbf{x}$ *in a suitable neighborhood of* $\mathbf{x}^*$*, then* $y(\mathbf{x})$ *has a local minimum at* $\mathbf{x} = \mathbf{x}^*$.

The proof of this theorem is by contradiction.

As will be seen, the method of steepest ascent (gradient search) is not an effective method even though it will converge to an optimum eventually. The algorithm tends to zigzag, and the rate of convergence is significantly slower than other algorithms. Consequently, the rate (or order) of convergence of an algorithm is another important theoretical property. The rate of convergence of a sequence $\mathbf{x}_k$ for an algorithm as described by Fletcher (4) is in terms of the norm of the difference of a point in the sequence $\mathbf{x}_k$ and the optimum x^* i.e., $\| \mathbf{x}_k - \mathbf{x}^* \|$.
If $\| \mathbf{x}_{k+1} - \mathbf{x}^* \| / \| \mathbf{x}_k - \mathbf{x}^* \| \rightarrow a$, then the rate of convergence is said to be linear or first-order if $a \geq 0$. It is said to be superlinear if $a = 0$. For an algorithm, it is desirable to have the value of a as small as possible. For some algorithms it is possible to show that $\| \mathbf{x}_{k+1} - \mathbf{x}^* \|^2 / \| \mathbf{x}_k - \mathbf{x}^* \|^2 \rightarrow a$, and for this case the rate of convergence is said to be quadratic or second-order. For the method of steepest ascent Fletcher (4) states that the rate of convergence is a slow rate of linear convergence that depends on the largest and smallest eigenvalues of the Hessian matrix.

Another criterion often used to compare algorithms is their ability to locate the optimum of quadratic functions. This is called *quadratic termination*. The justification for using this criterion for comparison is that near an optimum the function can be "adequately approximated by a quadratic form," according to Bazaraa and Shetty (56). They claim that an algorithm that does not perform well in minimizing a quadratic function probably will not do well for a general nonlinear function, especially near the optimum.

There are several caveats about relying on theoretical results in judging algorithms. One is that the existence of convergence and rate of convergence results for any algorithm does not a guarantee good performance in practice according to Fletcher (4). One reason is that these theoretical results do not account for computer round-off error that may be crucial. Both numerical experimentation with a variety of test functions and convergence, and rate of convergence proofs are required to give a reliable indication of good performance. Also, as discussed by Gill, et al. (6) conditions for achieving the theoretical rate of convergence may be rare since an infinite sequence does not exist on a computer. Moreover, the absence of a theorem on the rate of

convergence of an algorithm may be as much a measure of the difficulty of the proof as the inadequacy of the algorithm according to Gill, et al.(6).

Quasi-Newton Methods: These methods begin the search along a gradient line and use gradient information to build a quadratic fit to the economic model (profit function). Consequently, to understand these methods it is helpful to discuss the gradient search algorithm and Newton's method as background for the extension to the quasi-Newton algorithms. All of the algorithms involve a line search given by the following equation.

$$\mathbf{x}_{k+1}=\mathbf{x}_k - \alpha \nabla \boldsymbol{H}_k\, y(\mathbf{x}_k) \tag{5-2}$$

For gradient search $\mathbf{H}_k$ is **I**, the unit matrix; and α is the parameter of the gradient line. For Newton's method $\mathbf{H}_k$ is the inverse of the Hessian matrix, $\mathbf{H}^{-1}$; and α is one. For quasi-Newton methods $\mathbf{H}_k$ is a series of matrices beginning with the unit matrix, **I**, and ending with the inverse of the Hessian matrix, $\mathbf{H}^{-1}$. The quasi-Newton algorithm that employs the BFGS (Broyden, Fletcher, Golfarb, Shanno) formula for up-dating the Hessian matrix is considered to be the most effective of the unconstrained multivariable search techniques according to Fletcher (5). This formula is an extension of the DFP (Davidon, Fletcher, Powell) formula.

Gradient Search: Gradient search or the method of steepest ascent was presented in Chapter 2 as an example of the application of the method of Lagrange multipliers. However, let us consider briefly another approach to obtain this result that should give added insight to the method. First, the profit function, y(**x**), is expanded around point $\mathbf{x}_k$ in a Taylor series with only first order terms as:

$$\text{maximize: } \boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{x}_k) + \sum_{j=1}^{n} \frac{\partial \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_j}(\boldsymbol{x}_j - \boldsymbol{x}_{jk}) \tag{5-3}$$

In matrix notation, the above equation has the following form:

$$\text{maximize: } \boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{x}_k) + \nabla^T \boldsymbol{y}(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) \tag{5-4}$$

Then to maximize $y(\mathbf{x})$, the largest value of $\nabla \boldsymbol{y}(\boldsymbol{x}_k)^T(\boldsymbol{x} - \boldsymbol{x}_k)$ is to be used. When the largest value of $\nabla^T \boldsymbol{y}(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k)$ is determined, it has to be in the form of an equation that gives the way to change the individual x_j's to move in the direction of steepest ascent. This term can be written in vector notation as the dot product of two vectors.

$$\nabla \boldsymbol{y}(\boldsymbol{x}_k)^T(\boldsymbol{x} - \boldsymbol{x}_k) = \nabla \boldsymbol{y}(\boldsymbol{x}_k) \bullet (\boldsymbol{x} - \boldsymbol{x}_k) = \left|\nabla \boldsymbol{y}(\boldsymbol{x}_k)\right|\left|(\boldsymbol{x} - \boldsymbol{x}_k)\right| \cos\theta \tag{5-5}$$

The magnitude of the gradient of $y(\boldsymbol{x}_k)$ at point $\mathbf{x}_k$, $\left|\nabla \boldsymbol{y}(\boldsymbol{x}_k)\right|$, is known or can be measured at $\boldsymbol{x}_k$; and the magnitude of the vector $(\boldsymbol{x} - \boldsymbol{x}_k)$ is to be determined to maximize the dot product of the two vectors. In examining Equation 5-5, the largest value of the dot product is with the value

of $\boldsymbol{\theta} = 0$ where *cos* (0) = 1. Consequently, the two vectors $\nabla \boldsymbol{y}(\boldsymbol{x}_k)$ and ($\boldsymbol{x}$—$\boldsymbol{x}_k$) are collinear and are proportional. This is given by the following equation.

$$\boldsymbol{x} - \boldsymbol{x}_k = \alpha \nabla \boldsymbol{y}(\boldsymbol{x}_k) \tag{5-6}$$

where α is the proportionality constant and is also the parameter of the gradient line. Therefore, the gradient line, Equation 5-6 can be written as:

$$\boldsymbol{x} = \boldsymbol{x}_k + \alpha \nabla \boldsymbol{y}(\boldsymbol{x}_k) \tag{5-7}$$

The plus sign in Equation 5-7 indicates the direction of steepest ascent, and using a negative sign in the equation would give the direction of steepest descent. However, these directions are actually steep ascent (descent) rather than steepest ascent (descent). Only if the optimization problem is scaled such that a unit change in each of the independent variables produces the same change in the profit function will the gradient move in the direction of steepest ascent. The procedures for scaling have been described in detail by Wilde (10) and Wilde and Beightler (12), and scaling is a problem encountered with all search methods.

Comparing Equation 5-7 to Equation 5-2, it is seen that $\Delta \boldsymbol{H}_k = \boldsymbol{I}$, the identity matrix. An open-ended line search on α is required to locate the optimum along the gradient line.

The following short example illustrates the gradient method for a simple function with ellipsoidal contours. The zigzag behavior is observed as the algorithm moves from the starting point at (2, -2,1) to the minimum at (0,0,0) of a function that is the sum of squares.

Example 5–1

Search for the minimum of the following function using gradient search starting at point $\mathbf{x}_0 = (2,-2, 1)$.

$$y = 2x_1^2 + x_2^2 + 3x_3^2$$

The gradient line, Equation 6-7, for point $\mathbf{x}_0$ is:

$$\boldsymbol{x} = \boldsymbol{x}_0 + \alpha \nabla \boldsymbol{y}(\boldsymbol{x}_0)$$

and the three components of this equation are:

$$x_1 = x_{10} + \alpha \frac{\partial y(\mathbf{x}_0)}{\partial x_1}$$

$$x_2 = x_{20} + \alpha \frac{\partial y(\mathbf{x}_0)}{\partial x_2}$$

$$x_3 = x_{30} + \alpha \frac{\partial y(\underline{\mathbf{x}}_0)}{\partial x_3}$$

Evaluating the partial derivatives gives:

$\frac{\partial y}{\partial x_1} = 4x_1$ $\frac{\partial y(\mathbf{x}_0)}{\partial x_1} = 8$ $\frac{\partial y}{\partial x_2} = 2x_2$ $\frac{\partial y(\mathbf{x}_0)}{\partial x_2} = -4$ $\frac{\partial y}{\partial x_3} = 6x_3$ $\frac{\partial y(\mathbf{x}_0)}{\partial x_3} = 6$

The gradient line is:

$$x_1 = 2 + 8\alpha$$

$$x_2 = -2 - 4\alpha$$

$$x_3 = 1 + 6\alpha$$

Using the gradient line equations, $y(x_1,x_2,x_3)$ is converted into $y(\alpha)$ for an exact line search:

$$y = 2(2 + 8\alpha)^2 + (-2 - 4\alpha)^2 + 3(1 + 6\alpha)^2$$

and

$$\frac{dy}{d\alpha} = 32(2 + 8\alpha) - 8(-2 - 4\alpha) + 36(1 + 6\alpha) = 0 \rightarrow \alpha^* = -0.23016$$

Computing point $\boldsymbol{x}_1$ using $\alpha^* = -0.23016$ gives:

$$x_1 = 2 + 8(-0.23016) = 0.15872$$

$$x_2 = -2 - 4(-0.23016) = 1.0794$$

$$x_3 = 1 + 6(-0.23016) = -0.38096$$

Continuing, the partial derivatives are evaluated at $\mathbf{x}_1$ to give:

$\frac{\partial y}{\partial x_1}(\mathbf{x}_1) = 4(0.15892) = 0.63488$ $\frac{\partial y}{\partial x_2}(\mathbf{x}_1) = 2(1.0794) = 2.1588$

$\frac{\partial y}{\partial x_3}(\mathbf{x}_1) = 6(-0.38096) = -2.2858$

The gradient line at $\mathbf{x}_1$ is:

$$x_1 = 0.15872 + 0.63688\alpha$$

$x_2 = 1.0794 + 2.1588\alpha$

$x_3 = -0.38096 - 2.2858\alpha$

The value of α which minimizes $y(\alpha)$ along the gradient line from $\mathbf{x}_1$ is computed as was done previously, and the result is $\alpha^* = -0.2433$. Using this value of α the point $\mathbf{x}_2$ is computed as (0.004524, 0.5542, 0.1752). Then, the search is continued along the gradient line from $\mathbf{x}_2$ to $\mathbf{x}_3$. These results and those from subsequent application of the algorithm are tabulated below along with the previous results.

Iteration	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	α	y(**x**)
0	2	-2	1		15.0
				-0.23016	
1	0.1587	1.0794	-0.3810		1.6510
				-0.2433	
2	0.004254	0.5542	0.1752		0.3993
				-0.2568	
3	$-1.2\text{x}10^{-4}$	0.2696	-0.0947		0.09959
				-0.2436	
4	$3.4\text{x}10^{-6}$	0.1383	0.04371		0.02486
				-0.2568	
5	0	0.06727	-0.02365		0.006203
				-0.2435	
6	0	0.03452	0.01090		0.001548
				-0.2570	
7	0	$1.68\text{x}10^{-3}$	$-5.90\text{x}10^{-3}$		$2.8\text{x}10^{-4}$
				-0.4999	
8	0	$3.0\text{x}10^{-6}$	$1.0\text{x}10^{-6}$		$1.2\text{x}10^{-11}$

A stopping criterion, having the independent variables be less than or equal to $1\text{x}10^{-3}$, was used. Also, a criterion on the value of $y(\mathbf{x})$ could have been used.

Notice that the value of the parameter of the gradient line α is always negative. This indicates the algorithm is moving in the direction of steepest descent. As above results show, gradient search tends to take a zigzag path to the minimum of the function. This is typical of the performance of this algorithm.

Newton's Method: In the development of Newton's method, the Taylor series expansion of $y(\mathbf{x})$ about $\mathbf{x}_k$ includes the second order terms as shown below.

$$\text{optimize: } y(\boldsymbol{x}) = y(\boldsymbol{x}_k) + \sum_{j=1}^{n} \frac{\partial y(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_j}(\boldsymbol{x}_j - \boldsymbol{x}_{jk}) + \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 y(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j}(\boldsymbol{x}_i - \boldsymbol{x}_{ik})(\boldsymbol{x}_j - \boldsymbol{x}_{jk}) \qquad (5\text{-}8)$$

A more convenient way to write this equation is in matrix notation:

$$\text{optimize: } \boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{x}_k) + \nabla^T \boldsymbol{y}(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) + \tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}_k) \qquad (5\text{-}9)$$

where $\boldsymbol{H}$ is the Hessian matrix, the matrix of second partial derivatives evaluated at the point $\mathbf{x}_k$, and $(\mathbf{x} - \mathbf{x}_k)^T$ is the row vector which is the transpose of the column vector of the difference between the vector of independent variables $\mathbf{x}$ and the point $\mathbf{x}_k$ used for the Taylor series expansion.

The algorithm is developed by locating the stationary point of Equation 5-8 or 5-9 by setting the first partial derivatives with respect to $x_1, x_2, \ldots, x_n$ equal to zero. For Equation 5-8 the result is:

$$\begin{aligned} &\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_1} = \frac{\partial \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_1} + \sum_{j=1}^{n} \frac{\partial^2 \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_1 \partial \boldsymbol{x}_j}(\boldsymbol{x}_j - \boldsymbol{x}_{jk}) = 0 \\ &\vdots \\ &\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_n} = \frac{\partial \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_n} + \sum_{j=1}^{n} \frac{\partial^2 \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_n \partial \boldsymbol{x}_j}(\boldsymbol{x}_j - \boldsymbol{x}_{jk}) = 0 \end{aligned} \qquad (5\text{-}10)$$

which when written in terms of the Hessian matrix is:

$$\nabla y(\mathbf{x}_k) + \mathbf{H}(\mathbf{x} - \mathbf{x}_k) = 0 \qquad (5\text{-}11)$$

Then solving for $\mathbf{x}$, the optimum of the quadratic approximation, the following equation is obtained which is the Newton's method algorithm.

$$\mathbf{x} = \mathbf{x}_k - \mathbf{H}^{-1} \nabla y(\mathbf{x}_k) \qquad (5\text{-}12)$$

Comparing Equation 5-12 to Equation 5-2, it is seen that $\alpha = -1$ and $\mathbf{H}_k = \mathbf{H}^{-1}$, the inverse of the Hessian matrix. Also, a line search is not required for this method since $\alpha = -1$. However, more computational effort is required for one iteration of this algorithm than for one iteration of gradient search since the inverse of the Hessian matrix has to be evaluated in addition to the gradient vector. The same quadratic function of the gradient search algorithm example is used to illustrate Newton's method in the following example, and it shows the additional computations required.

Example 5-2

Search for the minimum of the function from Example 6-1 using Newton's method starting at point $\mathbf{x}_0 = (2, -2, 1)$.

$$y = 2x_1^2 + x_2^2 + 3x_3^2$$

From the previous example the gradient is:

$$\nabla y(\mathbf{x}_0)^T = (8, -4, 6)$$

The Hessian matrix formed from the second partial derivatives evaluated at $\mathbf{x}_0$ and its inverse is:

$$\boldsymbol{H}_0 = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 6 \end{bmatrix} \qquad \boldsymbol{H}_0^{-1} = \begin{bmatrix} 1/4 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/6 \end{bmatrix}$$

The algorithm is given by Equation 5-12, and for this example is:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ 1 \end{bmatrix} - \begin{bmatrix} 1/4 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/6 \end{bmatrix} \begin{bmatrix} 8 \\ -4 \\ 6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

The minimum of the quadratic function is located with one application of the algorithm.

In Newton's method, if $\mathbf{x}_k$ is not close to $\mathbf{x}^*$, it may happen that $\mathbf{H}^{-1}$ is not positive definite; and then the method may fail to converge in this case (26). However, if the starting point $\mathbf{x}_o$ is sufficiently close to a local optimum $\mathbf{x}^*$, the rate of convergence is second order as given by the following theorem from Fletcher (4).

If $\mathbf{x}_k$ is sufficiently close to $\mathbf{x}^*$ for some k, and if $\mathbf{H}^*$ is positive definite, then Newton's method is well defined for all k and converges at a second-order rate.

The proof of the theorem has $\mathbf{x}_k$ in the neighborhood of x^* and uses induction.

Newton's method has the property of quadratic termination as demonstrated by the example above. It arrives at the optimum of a quadratic function in a finite number of steps, one.

However, for nonlinear functions generally Newton's method moves methodically toward the optimum; but the computational effort required to compute the inverse of the Hessian matrix

at each iteration usually is excessive compared to other methods. Consequently, it is considered to be an inefficient middle game procedure for most problems.

Quasi-Newton Methods: To overcome these difficulties, quasi-Newton methods were developed which use the algorithm given by Equation 5-2. They begin with a search along the gradient line, and only gradient measurements are required for $\mathbf{H}_k$ in subsequent applications of the algorithm given by Equation 6-2. As the algorithm proceeds, a quadratic approximation to the profit function is developed only from the gradient measurements; and for a quadratic function of n independent variables, the optimum is reached after n applications of the algorithm.

Davidon developed the concept in 1959 and Fletcher and Powell in 1963 extended the methodology. As discussed by Fletcher (4) there have been a number of other contributors to this area, also. The DFP (Davidon, Fletcher, Powell) algorithm has become the best known of the quasi-Newton (variable metric or large-step gradient) algorithms. Some of its properties are superlinear rate of convergence on general functions, and quadratic termination using exact line searches on quadratic functions (4).

A number of variations of the functional form of the matrix $\mathbf{H}_k$ of Equation 5-2 with the properties described above have been developed, and some of these have been tabulated by Himmelblau (8). However, as previously stated, the BFGS algorithm that was developed in 1970 is preferable to the others; and this is currently well accepted according to Fletcher (4). The following paragraphs will describe the DFP and BFGS algorithms and illustrate each with an example. Convergence proofs and related information are given by Fletcher (4) and others (6, 7, 8, 9).

The DFP algorithm has the following form of Equation 5-2 for minimizing the function $y(\mathbf{x})$.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_{k+1} \mathbf{H}_k \nabla y(\mathbf{x}_k) \tag{5-13}$$

where α_{k+1} is the parameter of the line from $\mathbf{x}_k$ to locate $\mathbf{x}_{k+1}$ at the optimum, and $\mathbf{H}_k$ is given by the following equation (12).

$$\mathbf{H}_k = \mathbf{H}_{k-1} + \mathbf{A}_k + \mathbf{B}_k \tag{5-14}$$

The matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ are given by the following equations.

$$\boldsymbol{A}_k = \frac{(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})^T}{(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})^T (\nabla \boldsymbol{y}(\boldsymbol{x}_k - \nabla \boldsymbol{y}(\boldsymbol{x}_{k-1}))} \tag{5-15}$$

$$\boldsymbol{B}_k = \frac{-\boldsymbol{H}_{k-1}\left[\nabla \boldsymbol{y}(\boldsymbol{x}_k) - \nabla \boldsymbol{y}(\boldsymbol{x}_{k-1})\right]\left[\nabla \boldsymbol{y}(\boldsymbol{x}_k) - \nabla \boldsymbol{y}(\boldsymbol{x}_{k-1})\right]^T \boldsymbol{H}_{k-1}^T}{\left[\nabla \boldsymbol{y}(\boldsymbol{x}_k) - \nabla \boldsymbol{y}(\boldsymbol{x}_{k-1})\right]^T \boldsymbol{H}_{k-1}\left[\nabla \boldsymbol{y}(\boldsymbol{x}_k) - \nabla \boldsymbol{y}(\boldsymbol{x}_{k-1})\right]} \tag{5-16}$$

The algorithm begins with a search along the gradient line from the starting point $\mathbf{x}_0$ as given by the following equation obtained from Equation 5-13 with k = 0.

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha \mathbf{H}_0 \nabla y(\mathbf{x}_0) \tag{5-17}$$

where $\mathbf{H}_0 = \mathbf{I}$ is the unit matrix. This equation is the same as Equation 5-7 for the gradient line.

The algorithm continues using Equation 5-13 with updates using Equations 5-15 and 5-16 until a stopping criterion is met. However, for a quadratic function with *n* independent variables the method converges to the optimum after *n* iterations (quadratic termination) if exact line searches are used.

The matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ have been constructed so their sums would have the specific properties shown below (12).

$$\sum_{k=0}^{n} \boldsymbol{A}_k = \boldsymbol{H}^{-1} \tag{5-18}$$

$$\sum_{k=0}^{n} \boldsymbol{B}_k = -\boldsymbol{H}_0 = -\boldsymbol{I} \tag{5-19}$$

The sum of the n matrices $\mathbf{A}_k$ generates the inverse of the Hessian matrix $\mathbf{H}^{-1}$ to have Equation 5-13 be the same as Newton's method, Equation 5-12, at the end of n iterations. The sum of the matrices $\mathbf{B}_k$ generates the negative of the unit matrix $\mathbf{I}$ at the end of n iterations to cancel the first step of the algorithm when $\mathbf{I}$ was used for $\mathbf{H}_0$ in Equation 5-17.

The development of the algorithm and the proofs for the rate of convergence and quadratic termination are given by Fletcher (4). Also, the procedure is applicable to and effective on nonlinear functions. According to Fletcher (4) for general functions it preserves positive definite $\mathbf{H}_k$ matrices, and thus the descent property holds. Also, it has a superlinear rate of convergence, and it converges to the global minimum of strictly convex functions if exact line searches are used.

The following example illustrates the use of the DFP algorithm for a quadratic function with three independent variables. Consequently, the optimum is reached with three applications of the algorithm.

Example 5-3 (14)

Determine the minimum of the following function using the DFP algorithm starting at $\mathbf{x}_0^T = (0,0,0)$.

$$\text{minimize: } 5x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3 - 2x_1x_3 - 6x_3$$

Performing the appropriate partial differentiation, the gradient vector $\nabla y(\mathbf{x})$ and the Hessian matrix are:

$$\nabla y(\mathbf{x}) = \qquad \nabla \boldsymbol{y}(\boldsymbol{x}) = \begin{bmatrix} 10\boldsymbol{x}_1 + 2\boldsymbol{x}_2 - 3\boldsymbol{x}_3 \\ 2\boldsymbol{x}_1 + 4\boldsymbol{x}_2 + 2\boldsymbol{x}_3 \\ -2\boldsymbol{x}_1 + 2\boldsymbol{x}_2 + 4\boldsymbol{x}_3 - 6 \end{bmatrix} \qquad \boldsymbol{H} = \begin{bmatrix} 10 & 2 & -2 \\ 2 & 4 & 2 \\ -2 & 2 & 4 \end{bmatrix}$$

Using Equation 6-17 to start the algorithm gives:

$$\begin{bmatrix} \boldsymbol{x}_{11} \\ \boldsymbol{x}_{21} \\ \boldsymbol{x}_{31} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \alpha_1 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 6\alpha_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3/2 \end{bmatrix}$$

The optimal value of α_1 was determined by an exact line search with Equation 6-17 using $x_1 = 0$, $x_2 = 0$, $x_3 = 6\alpha_1$ as follows.

$$y(\alpha_1) = 2(6\alpha_1)^2 - 6(6\alpha_1) = 72\alpha_1^2 - 36\alpha_1$$

$$dy/\mathrm{d}\,\alpha_1 = 144\alpha_1 - 36 = 0 \rightarrow \alpha_1 = ¼$$

The value of $\mathbf{x}_1$ is computed by substituting for α_1 in the previous equation.

$$\mathbf{x}_1{}^T = (0, 0, 3/2) \qquad \nabla y(\mathbf{x}_1)^T = (-3, 3, 0) \quad \nabla y(\mathbf{x}_0)^T = (0, 0, -6)$$

The algorithm continues using Equations 5-13 and 5-14 for k=1.

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha_2 \mathbf{H}_1 \nabla y(\mathbf{x}_1)$$

or

$$\begin{bmatrix} \boldsymbol{x}_{12} \\ \boldsymbol{x}_{22} \\ \boldsymbol{x}_{32} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3/2 \end{bmatrix} - \alpha_2 \begin{bmatrix} 5/6 & 1/6 & 1/3 \\ 1/6 & 5/6 & -1/3 \\ 1/3 & -1/3 & 7/12 \end{bmatrix} \begin{bmatrix} -3 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 2\alpha_2 \\ -2\alpha_2 \\ 3/2 + 2\alpha_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 5/2 \end{bmatrix}$$

where α_2 is determined by an exact line search as shown below.

$$\mathbf{H}_1 = \mathbf{H}_0 + \mathbf{A}_1 + \mathbf{B}_1$$

and $\mathbf{A}_1$ and $\mathbf{B}_1$ are given by Equations 5-15 and 5-16.

$$A_1 = \frac{\begin{bmatrix} 0 \\ 0 \\ 3/2 \end{bmatrix}\begin{bmatrix} 0 & 0 & 3/2 \end{bmatrix}}{\begin{bmatrix} 0 & 0 & 3/2 \end{bmatrix}\begin{bmatrix} -3 \\ 3 \\ 6 \end{bmatrix}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1/4 \end{bmatrix}$$

$$B_1 = -\frac{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} -3 \\ 3 \\ 6 \end{bmatrix}\begin{bmatrix} -3 & 3 & 6 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}{\begin{bmatrix} -3 & 3 & 6 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} -3 \\ 3 \\ 6 \end{bmatrix}} = \begin{bmatrix} -1/6 & 1/6 & 1/3 \\ 1/6 & -1/6 & -1/3 \\ 1/3 & -1/3 & -2/3 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1/4 \end{bmatrix} + \begin{bmatrix} -1/6 & 1/6 & 1/3 \\ 1/6 & -1/6 & -1/3 \\ 1/3 & -1/3 & -2/3 \end{bmatrix} = \begin{bmatrix} 5/6 & 1/6 & 1/3 \\ 1/6 & 5/6 & -1/3 \\ 1/3 & -1/3 & 7/12 \end{bmatrix}$$

The optimal value of α_2 is determined by an exact line search as follows.

$y(\alpha_2) = 12\alpha_2^2 - 12\alpha_2 + 9/2$ $\qquad$ $dy / d\alpha_2 = 24\alpha_2 - 12 = 0 - \alpha_2 = ½$

The value of $\mathbf{x}_2$ is computed by substituting for α_2 in the previous equation.

$\mathbf{x}_2^T = (1, -1, 5/2)$ $\qquad$ $\nabla y(\mathbf{x}_2)^T = (3, 3, 0)$

The computation of $\mathbf{x}_3$ uses Equations 5-13 and 5-14 as follows:

$$\mathbf{x}_3 = \mathbf{x}_2 - \alpha_3 \mathbf{H}_2 \nabla y(\mathbf{x}_2)$$

and

$$H_2 = H_1 + A_2 + B_1 = \begin{bmatrix} 1/16 & -1/6 & 1/16 \\ -1/6 & 29/30 & -17/30 \\ 1/6 & -17/30 & 37/60 \end{bmatrix}$$

where

$$A_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} \Big/ \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/6 & -1/6 & 1/6 \\ -1/6 & 1/6 & -1/6 \\ 1/6 & -1/6 & 1/6 \end{bmatrix}$$

$$B_2 = \frac{\begin{bmatrix} 5/6 & 1/6 & 1/3 \\ 1/6 & 5/6 & -1/3 \\ 1/3 & -1/3 & 7/12 \end{bmatrix} \begin{bmatrix} 6 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 6 & 0 & 0 \end{bmatrix} \begin{bmatrix} 5/6 & 1/6 & 1/3 \\ -1/6 & 1/6 & -1/3 \\ 1/6 & -1/6 & 1/6 \end{bmatrix}}{\begin{bmatrix} 6 & 0 & 0 \end{bmatrix} \begin{bmatrix} 5/6 & 1/6 & -1/3 \\ 1/6 & 5/6 & -1/3 \\ 1/3 & -1/3 & 7/12 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix}} = \begin{bmatrix} -5/6 & -1/6 & -1/3 \\ -1/6 & -1/30 & -1/15 \\ -1/3 & -1/15 & -2/15 \end{bmatrix}$$

$$\begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 5/2 \end{bmatrix} - \alpha_3 \begin{bmatrix} 1/6 & -1/6 & 1/6 \\ -1/6 & 29/30 & -17/30 \\ 1/6 & 17/30 & 37/30 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1-12\alpha_3/5 \\ 5/2+6\alpha_3/5 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}$$

The optimal value of α_3 was determined by an exact line search as follows:

$$y(\alpha_3) = 5 + 2(1 + 12\alpha_3/5)^2 + 2(5/2 + 6\alpha_3/5)^2 - 2(1 + 12\alpha_3/5) \\ -2(1 + 12\alpha_3/5)(5/2 + 6\alpha_3/5) - 2(5/2 + 6\alpha_3/5) - 6(5/2 + 6\alpha_3/5)$$

Setting $dy(\alpha_3)/d\alpha_3 = 0$ and solving for α_3 gives $\alpha_3 = 5/12$ and $\mathbf{x}_3^T = (1, -2, 3)$ which is the value of the function at the minimum.

In the preceding example exact line searches were used to have the DFP algorithm proceed to the optimum. However, in optimization problems encountered in industrial practice exact line searches are not possible; and numerical single variable search methods must be used, ones such as golden section search or the quadratic method. However, the previously mentioned BFGS method will converge to the optimum of a convex function even when inexact line searches are used. Also, this global convergence property has not been demonstrated for other algorithms like the *DFP* algorithm according to Fletcher (4). Consequently, this may be part of the reason that the *BFGS* algorithm has demonstrated generally more satisfactory performance than other methods in numerical experiments, even though it is a more elaborate formula. The BFGS matrix up-date formula comparable to Equations (5-14), (5-15) and (5-16) as given by Fletcher (4) is:

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k - \left[\frac{\boldsymbol{H}_k \gamma_k \delta_k^T + \delta_k \gamma_k^T \boldsymbol{H}_k}{\delta_k^T \gamma_k} \right] + \left[1 + \frac{\gamma_k^T \boldsymbol{H}_k \gamma_k}{\delta_k^T \gamma_k} \right] \left[\frac{\delta_k \delta_k^T}{\delta_k^T \gamma_k} \right]$$

$$\delta_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \qquad (5\text{-}20)$$

$$\gamma_k = \nabla \boldsymbol{y}(\boldsymbol{x}_{k+1}) - \nabla \boldsymbol{y}(\boldsymbol{x}_k)$$

This equation is used in place of Equation 6-14 in the algorithm given by Equation 5-13. The procedure is the same in that a search along the gradient line from starting point $\mathbf{x}_0$ is conducted initially according to Equation 5-17. Then the Hessian matrix is updated using Equation 5-20, and for quadratic functions the method arrives at the minimum after n iterations. The following example illustrates the procedure for the BFGS algorithm using the function of Example 5-3.

Example 5-4 (14)

Determine the minimum of the following function using the BFGS algorithm starting at $\mathbf{x}_0$ = (0,0,0).

$$\text{Minimize: } 5x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3 - 2x_1x_3 - 6x_3$$

The first application of the algorithm is the same as Example 5-3 that is a search along the gradient line through $\mathbf{x}_0$ = (0,0,0). These results were:

$\mathbf{x}_1^T = (0, 0, 3/2)$ $\qquad\qquad$ $\nabla y(\mathbf{x}_1)^T = (-3, 3, 0)$

$\mathbf{x}_0^T = (0, 0, 0)$ $\qquad\qquad$ $\nabla y(\mathbf{x}_0)^T = (0, 0, -6)$

The algorithm continues using Equations 5-13 and 5-20 for k=1.

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha_2 \mathbf{H}_1 \nabla y(\mathbf{x}_1)$$

or

$$\begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3/2 \end{bmatrix} - \alpha_2 \begin{bmatrix} 1 & 0 & 1/2 \\ 0 & 1 & -1/2 \\ 1/2 & -1/2 & 3/4 \end{bmatrix} \begin{bmatrix} -3 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 3\alpha_2 \\ -3\alpha_2 \\ 3/2 + 3\alpha_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 5/2 \end{bmatrix}$$

where

$$\boldsymbol{H}_1 = \boldsymbol{H}_0 - \left[\frac{\boldsymbol{H}_0\gamma_0\delta_0^T + \delta_0\gamma_0^T\boldsymbol{H}_0}{\delta_k^T\gamma_k}\right] + \left[1 + \frac{\gamma_0^T\boldsymbol{H}_k\gamma_k}{\delta_0^T\gamma_0}\right]\left[\frac{\delta_0\delta_0^T}{\delta_0^T\gamma_0}\right]$$

$$\boldsymbol{H}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \delta_0 = \boldsymbol{x}_1 - \boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 3/2 \end{bmatrix} \quad \gamma_0 = \nabla\boldsymbol{y}(\boldsymbol{x}_1) - \nabla\boldsymbol{y}(\boldsymbol{x}_0) = \begin{bmatrix} -3 \\ 3 \\ 6 \end{bmatrix}$$

$$\delta_0^T\gamma_0 = 9 \qquad \gamma_0^T\boldsymbol{H}_0\gamma_0 = 54$$

$$\boldsymbol{H}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & -1/2 \\ 0 & 0 & 1/2 \\ -1/2 & 1/2 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 7/4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1/2 \\ 0 & 1 & -1/2 \\ 1/2 & -1/2 & 3/4 \end{bmatrix}$$

The optimal value of α_2 is determined by an exact line search using $x_1 = 3\alpha_2$, $x_2 = -3\alpha_2$, $x_3 = 3/2 + 3\alpha_2$ in the function being minimized to give:

$$y = 27\alpha_2^2 - 18\alpha_2 + 4½ \qquad dy/d\alpha_2 = 54\alpha_2 - 18 = 0 \rightarrow \alpha_2 = 1/3$$

The value for $\mathbf{x}_2$ is computed by substituting for α_2 in the previous equation.

$$\mathbf{x}_2^T = (1, -1, 5/2) \qquad \nabla^T y(\mathbf{x}_2) = (3, 3, 0)$$

The computation of $\mathbf{x}_3$ repeats the application of the algorithm as follows:

$$\mathbf{x}_3 = \mathbf{x}_2 - \alpha_3 \mathbf{H}_2 \nabla y(\mathbf{x}_2)$$

or

$$\begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 5/2 \end{bmatrix} - \alpha_2 \begin{bmatrix} 1/6 & -1/6 & 1/6 \\ -1/6 & 13/6 & -7/6 \\ 1/6 & -7/6 & 11/12 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1-6\alpha_3 \\ 5/2+3\alpha_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}$$

where

$$\boldsymbol{\delta}_1^T = (1,-1,1) \quad \boldsymbol{\gamma}_1^T = (6,0,0) \quad \boldsymbol{\delta}_1^T \boldsymbol{\gamma}_1 = 6 \quad \boldsymbol{\gamma}_1^T \mathbf{H}_1 \boldsymbol{\gamma}_1 = 36$$

$$\boldsymbol{H}_2 = \begin{bmatrix} 1 & 0 & 1/2 \\ 0 & 1 & -1/2 \\ 1/2 & -1/2 & 3/4 \end{bmatrix} - \begin{bmatrix} 2 & -1 & 3/2 \\ -1 & 0 & -1/2 \\ 3/2 & -1/2 & 1 \end{bmatrix} + \begin{bmatrix} 7/6 & -7/6 & 7/6 \\ -7/6 & 7/6 & -7/6 \\ 7/6 & -7/6 & 7/6 \end{bmatrix} = \begin{bmatrix} 1/6 & -1/6 & 1/6 \\ -1/6 & 13/6 & -7/6 \\ 1/6 & -7/6 & 11/12 \end{bmatrix}$$

The optimal value of α_3 is determined by an exact line search using $x_{13} = 1$, $x_{23} = -1-6\alpha_3$, $x_{33} = 5/2 + 3\alpha_2$ in the function being minimized to give $y(\alpha_3)$. The value of $\alpha_3 = 1/6$ is determined as previously by setting $dy(\alpha_3)/d\alpha_3 = 0$, and the optimal value of $\mathbf{x}_3^T = (1,-2,3)$ is computed which is the value of the function at the minimum.

A program for the BFGS method is given in Table 5-4 at the end of this chapter. It employs the Fibonacci search program described in Chapter 5 for the line searches. This method and the program are applicable to functions that are not quadratic, also. However, the property of quadratic termination to the optimum in a predetermined number of steps is applicable to quadratic functions only; and a stopping criterion has to be specified for general nonlinear functions. In this program

the function to be minimized and the stopping criterion, EPS, are to be supplied by the user; and the program terminates when the magnitude of successive values of the profit function are less than the value of the stopping criterion. The solution to the problem of Example 6-4 is given to illustrate the use of the program.

Conjugate Gradient and Direction Methods: The distinguishing feature of these methods is that they have the quadratic termination property. The conjugate direction methods do not require derivative measurements, and the conjugate gradient methods only require gradient measurements. These procedures have been effective on a number of optimization problems, and they have been summarized by Fletcher (4) and others (6, 7, 8, 9, 15). The conjugate gradient and direction algorithms can locate the optimum of a quadratic function by searching only once along conjugate directions if exact line searches are used (quadratic termination), and all methods rely on the theorem given below. They differ in the way the conjugate directions are generated, and the objective has been to develop efficient methods for general functions (4). Two methods that have been consistently better performers than the others will be described, Powell's method for conjugate directions and gradient partan for conjugate gradients.

The idea for these methods is based on the fact that the optimum of a function that is separable can be found by optimizing separately each component. A quadratic function can be converted into a separable function, a sum of perfect squares (15), using a linear transformation; and the optimum can be found by a single variable search on each of the n transformed independent variables. The directions from the transformations are called *conjugate directions*.

A quadratic function to be optimized can have the following form.

$$y(\mathbf{x}) = a + \mathbf{b}^T \mathbf{x} + \mathbf{x}^T \mathbf{H}\mathbf{x} \tag{5-21}$$

Then using of the properties of a quadratic function, e.g. **H** is a positive definite, symmetric matrix, it can be shown that a set of linearly independent vectors $\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n$; are mutually conjugate with respect to **H** if:

$$\mathbf{s}_i^T \mathbf{H}\, \mathbf{s}_j = 0 \tag{5-22}$$

Then using this property, sets of conjugate search directions can be constructed that minimize the quadratic function, Equation 5-21, as illustrated by Himmelblau (8). The theorem on which these methods rely, as given by Fletcher (4), is:

A conjugate direction method terminates for a quadratic function in at most n exact line searches, and each $\mathbf{x}_{i+1}$ *is the minimizer in the subspace generated by* $\mathbf{x}_i$ *and the directions* $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_i$ *(that is the set of points* $\left\{ \boldsymbol{x} \mid \boldsymbol{x} = \boldsymbol{x}_1 + \sum_{j=1}^{i} \alpha_j \boldsymbol{s}_j \forall \alpha_j \right\}$).

The proof uses the stationary point necessary conditions, Equation 5-22 and the fact that mutually conjugate vectors are linearly independent (4, 9, 26, 57). However, the proof does not give insight into the means of constructing conjugate directions (4).

The notion of conjugate directions is a generalization of orthogonal directions where **H** = **I** in Equation 5-22 according to Avriel (9); and algorithms, such as Powell's method, initially search along orthogonal coordinate axes. Also, the DFP and the BFGS methods are conjugate direction methods when exact line searches are used (7).

Searching along conjugate directions can be represented by the following equation.

$$\boldsymbol{x} = \boldsymbol{x}_0 + \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i \tag{5-23}$$

where α_i is the parameter of the line in the conjugate directions (the orthogonal coordinate axes initially in Powell's method), and $\mathbf{x}_i$ is the vector that gives the conjugate directions (a coordinate axis e.g. $\mathbf{x}_i = (0, \ldots, 0, \mathbf{x}_i, 0, \ldots 0)$ in Powell's method). For a given direction of search, $\mathbf{x}_i$, the value of α_i is located to give the optimum of $y(\mathbf{x}_i)$ along the line of search. The function to be optimized can be written as:

$$\boldsymbol{y}(\boldsymbol{x}^*) = \boldsymbol{y}(\boldsymbol{x}_0 + \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i) \tag{5-24}$$

Then to locate the optimum, $\mathbf{x}^*$, an exact line search is conducted on each of the α_i's individually. The optimum of $y(\mathbf{x})$ is then determined by exact line searches in each of the conjugate directions. Further details are given by Fletcher (4), Avriel (9), and Powell (57) about the theory for these methods.

The two methods most frequently associated with conjugate direction are illustrated in Figure 5-1. These are Powell's method (57) and steep ascent partan (12). In Powell's method, the conjugate directions are the orthogonal coordinate axes initially, and in steep ascent partan the conjugate directions are the gradient lines. Also, both procedures employ an acceleration step. In the following paragraphs these two methods are discussed in more detail for n independent variables and are illustrated with an example.

In Powell's algorithm (9) the procedure begins at a starting point $\mathbf{x}_0$, and each application of the algorithm consists of (n+2) successive exact line searches. The first (n + 1) line searches are along each of the n coordinate axes. The (n+2)nd line search goes from the best point obtained from the first line search through the best point obtained at the end of the (n+1) line searches. If the function is quadratic, this will locate the optimum. If it is not, then the search is continued with one of the first n direction replaced by the (n + 1)th direction; and the procedure is repeated until a stopping criterion is met. This is illustrated in Figure 5-1(a) for two independent variables.

The basic procedure for an iteration as given by Powell (57) is as follows for a function of n independent variables starting at initial point $\mathbf{x}_1$ with the conjugate direction $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$ chosen as the coordinate axes.

Powell's Method for a General Function (57)

0. Calculate α_1 so that $y(\mathbf{x}_I + \alpha_I \mathbf{s}_n)$ is a minimum, and define $\mathbf{x}_0 = \mathbf{x}_I + \alpha_I \mathbf{s}_n$.
1. For $j = 1, 2, ..., n$:
   Calculate α_j so that $y(\mathbf{x}_{j-1} + \alpha_j \mathbf{s}_j)$ is a minimum.
   Define $\mathbf{x}_j = \mathbf{x}_{j-1} + \alpha_j \mathbf{s}_j$.
   Replace $\mathbf{s}_j$ with $\mathbf{s}_{j+1}$.
2. Replace $\mathbf{s}_n$ with $\mathbf{x}_n - \mathbf{x}_0$.
3. Choose α so that $y[\mathbf{x}_0 + \alpha(\mathbf{x}_n - \mathbf{x}_0)]$ is a minimum, and replace $\mathbf{x}_0$ with $\mathbf{x}_0 + \alpha(\mathbf{x}_n - \mathbf{x}_0)$.
4. Repeat steps 1-3 until a stopping criterion is met.

For a quadratic function the method will arrive at the minimum on completing Step 3. For a general function Steps 1-3 are repeated until a stopping criterion is satisfied. Step 0 is required to start the method by having $\mathbf{x}_0$, the point beginning the iteration steps 1-3, be a minimum point on the contour tangent line $\mathbf{s}_n$. The following example illustrates the above procedure for a quadratic function with two independent variables.

Example 5-5 (8)

Determine the minimum of the following function using Powell's method starting at initial point $\mathbf{x}_I = (2,2)$.

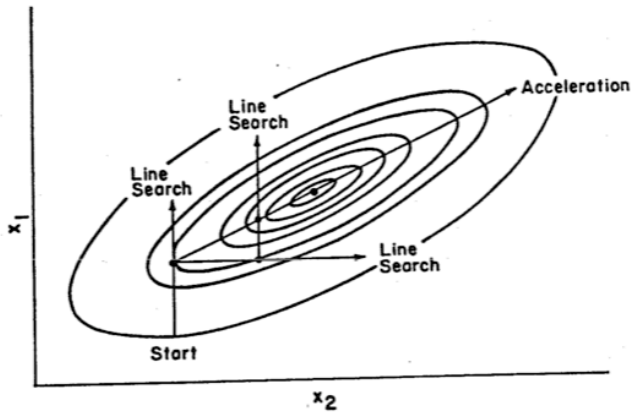
$$\text{minimize: } y = 2x_1^2 + x_2^2 - x_1x_2$$

As shown in Figure 5-2, the procedure begins at point $\mathbf{x}_I = (2, 2)$, and step 0 locates the minimum on the contour tangent line $\mathbf{s}_n$, $\mathbf{x}_0$, by a single variable search along coordinate axis n (= 2) as follows:

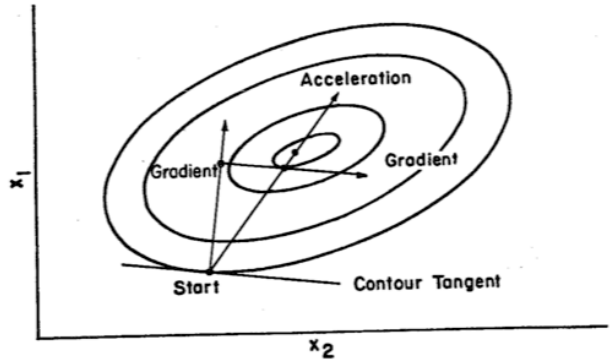
Step 0. $n = 2$ $\mathbf{s}_1^T = (1, 0)$ $\mathbf{s}_2^T = (0,1)$ $\mathbf{x}_I^T = (2, 2)$

$$\mathbf{x}_0 = \mathbf{x}_I + \alpha_I \mathbf{s}_2 \quad \text{or} \quad \begin{bmatrix} x_{1,0} \\ x_{2,0} \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \alpha_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$y(\alpha_I) = 2(2)^2 + (2 + \alpha_I)^2 - (2)(2 + \alpha_I)$$



a) Powell's Method



b) Steep Ascent Partan

Figure 5-1 Graphical Illustration of Powell's Method and Steep Ascent Partan

Using an exact line search, $\alpha_1 = -1$ and $\mathbf{x}_0^T = (2, 1)$.

Step 1. $\mathbf{s}_1^T = (1,0)$ $\mathbf{s}_2^T = (0,1)$ $\mathbf{x}_0^T = (2, 1)$

$j = 1$ $\mathbf{x}_1 = \mathbf{x}_0 + \alpha_1 \mathbf{s}_1$ or $\begin{bmatrix} x_{1,1} \\ x_{2,1} \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$y(\alpha_1) = 2(2 + \alpha_1)^2 + (1)^2 - (2 + \alpha_1)(1)$

Using an exact line search, $\alpha_1 = -7/4$ and $\mathbf{x}_1^T = (¼, 1)$. Replace $\mathbf{s}_1$ with $\mathbf{s}_2$

$$j = 2 \qquad \mathbf{x}_2 = \mathbf{x}_1 + \alpha_2 \mathbf{s}_2 \qquad \text{or} \quad \begin{bmatrix} x_{1,2} \\ x_{2,2} \end{bmatrix} = \begin{bmatrix} 1/4 \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$y(\alpha_2) = 2(¼)^2 + (1 + \alpha_2)^2 - (¼)(1 + \alpha_2)$$

Using an exact line search, $\alpha_2 = -7/8$ and $\mathbf{x}_2^T = (1/4, 1/8)$

Step 2. $\mathbf{s}_2$ is replaced with $\mathbf{x}_2 - \mathbf{x}_0 = \begin{bmatrix} 1/4 \\ 1/8 \end{bmatrix} - \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -1\ 3/4 \\ -7/8 \end{bmatrix}$

Step 3. Choose α_3 so that $y[\mathbf{x}_2 + \alpha_3(\mathbf{x}_2 - \mathbf{x}_0)]$ is a minimum. Let

$$\mathbf{x}_3 = \mathbf{x}_2 + \alpha_3(\mathbf{x}_2 - \mathbf{x}_0) = \begin{bmatrix} 1/4 \\ 1/8 \end{bmatrix} + \alpha \begin{bmatrix} -1\ 3/4 \\ -7/8 \end{bmatrix}$$

$$y(\alpha_3) = 2(1/4 - 1\ 3/4\alpha_3)^2 + (1/8 - 7/8\alpha_3)^2 - (1/4 - 1\ 3/4\alpha_3)(1/8 - 7/8\alpha_3)$$

Using an exact line search, $\alpha_3 = 1/7$ and $\mathbf{x}_3^T = (0, 0)$. $\mathbf{x}_3$ is the minimum of the quadratic function, and the procedure ends.

If the function in the above example had not been quadratic, the procedure would have continued using $\mathbf{s}_1^T = (0, 1)$ and $\mathbf{s}_2^T = (-1\ 3/4, -7/8)$, i.e. the direction $(\mathbf{x}_2 - \mathbf{x}_0)$ for the second cycle. In the third cycle, $\mathbf{s}_1$ would be replaced by $\mathbf{s}_2$ and $\mathbf{s}_2$ would be replaced by the new acceleration direction. The cycles are repeated until a stopping criterion is met.

Powell (57) has pointed out that this procedure required modification if the acceleration directions become close to being linearly dependent. He reported that this possibility has been found to be serious if the function depended on more than five variables. Powell developed a test that determined if the new conjugate direction was to replace one of the existing directions or if the iterative cycle, steps 1-3, was to be repeated with the existing set of linearly independent directions. If the reader plans to use this procedure Powell's paper (57) should be examined for the details of this test which was said to be essential to minimize a function of twenty independent variables.

Powell's method has been called one of the more efficient and reliable of the direct search methods (15). The reason is its relative simplicity and quadratic termination property. The method uses sectioning and does not employ the acceleration step. It just searches along the coordinate axes one at a time and can be confounded by resolution ridges.

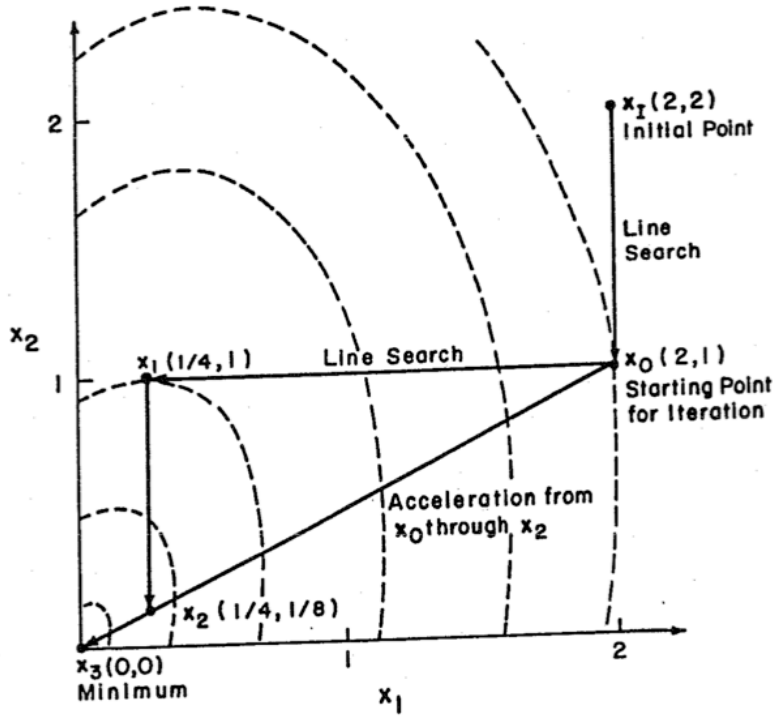


Figure 5-2 Illustration of Powell's Method for $y = x_1^2 + 2x_2^2 - x_1x_2$ from Example 5-5 after Himmelblau (8)

The conjugate gradient method, gradient partan, has proved to be as effective as Powell's method. It is an extension of gradient search and has the ability to locate the optimum of a function with ellipsoidal contours (quadratic termination) in a finite number of steps. The term partan comes from a class of search techniques that employ parallel tangents (12). These methods move in conjugate directions; or in the case of gradient partan, they move in the direction of conjugate gradients. The procedure is diagrammed in Figure 5-1 (b), and this shows that the gradient line is perpendicular to the contour tangent. Thus, the method can begin directly from the starting point as described below.

For two variables the procedure employs two gradient searches followed by an acceleration step, as shown in Figure 5-1 (a), for a function with elliptical contours. The acceleration line passes through the optimum. The equations for the gradient and acceleration lines for this method are:

Gradient line: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \nabla y(\mathbf{x}_k)$ (5-25)

Acceleration: $\mathbf{x}_{k+1} = \mathbf{x}_{k-3} + \alpha (\mathbf{x}_k - \mathbf{x}_{k-3})$ (5-26)

For more than two variables the diagram below shows the sequence of gradient searches and acceleration steps required for a function with ellipsoidal contours.

Gradient Partan Algorithm for a Function with Ellipsoidal Contours
Number of Independent Variables

	2	**3**	**4**	**n**
Start: $\mathbf{x}_0$				
Gradient:	$\mathbf{x}_0 \rightarrow \mathbf{x}_2$			
Gradient:	$\mathbf{x}_2 \rightarrow \mathbf{x}_3$	$\mathbf{x}_4 \rightarrow \mathbf{x}_5$	$\mathbf{x}_6 \rightarrow \mathbf{x}_7$	$\mathbf{x}_{2n-2} \rightarrow \mathbf{x}_{2n-1}$
Accelerate:	$\mathbf{x}_0 \rightarrow \mathbf{x}_3 \rightarrow \mathbf{x}_4$	$\mathbf{x}_2 \rightarrow \mathbf{x}_5 \rightarrow \mathbf{x}_6$	$\mathbf{x}_4 \rightarrow \mathbf{x}_7 \rightarrow \mathbf{x}_8$	$\mathbf{x}_{2n-4} \rightarrow \mathbf{x}_{2n-1} \rightarrow \mathbf{x}_{2n}$

To have the recursion relation shown above, it is necessary to omit a point numbered $\mathbf{x}_1$.

As shown in the above diagram for a function of *n* independent variables with ellipsoidal contours, a total of *n* gradient measurements and (2n-1) exact line searches are required to arrive at the optimum point $\mathbf{x}_{2n}$. The search begins at $\mathbf{x}_0$, and a search along the gradient line locates point $\mathbf{x}_2$. This is followed by another search along the gradient line to arrive at point $\mathbf{x}_3$. Then an acceleration step is performed from point $\mathbf{x}_0$ through $\mathbf{x}_3$ to arrive at point $\mathbf{x}_4$, the optimum of a function with elliptical contours. For *n* independent variables the procedure continues by repeating gradient searches and accelerations to arrive at point $\mathbf{x}_{2n}$, the optimum of a function of *n* independent variables having ellipsoidal contours. This procedure is illustrated in the following example for a function with three independent variables. In this case the optimum will be reached with three gradient measurements and five line searches.

Example 5-6 (10)

Determine the minimum of the following function using gradient partan starting at the point $\mathbf{x}_0 = (2, -2, 1)$

$$y = 2x_1^2 + x_2^2 + 3x_3^3$$

Beginning with a gradient search from point $\mathbf{x}_0$ to point $\mathbf{x}_2$, Equation 5-7 is used.

$$\mathbf{x} = \mathbf{x}_0 + \alpha \nabla y(\mathbf{x}_0)$$

or

$$x_1 = 2 + 8\alpha$$

$$x_2 = -2 - 4\alpha \quad \text{where } \nabla y = \begin{bmatrix} \partial y/\partial x_1 \\ \partial y/\partial x_{21} \\ \partial y/\partial x_{31} \end{bmatrix} = \begin{bmatrix} 4x_1 \\ 2x_2 \\ 6x_3 \end{bmatrix}$$

$$x_3 = 1 + 6\alpha$$

Performing an exact line search along the gradient from $\mathbf{x}_0$ gives:

$$y = 2(2 + 8\alpha)^2 + (-2 - 4\alpha)^2 + 3(1 + 6\alpha)^2$$

Setting $dy/d\alpha = 0$ to locate the minimum of y along the gradient line gives:

$$\frac{dy}{d\alpha} = 32(2 + 8\alpha) - 8(-2 - 4\alpha) + 36(1 + 6\alpha) = 0$$

Solving for the optimum value of α gives $\alpha^* = -0.2302$. Using α^* to compute $\mathbf{x}_2$ gives $(0.1584, -1.079, -0.3810)^T$, and the gradient line at $\mathbf{x}_2$ is:

$$x_1 = 0.1584 + 0.6336\alpha$$

$$x_2 = -1.079 - 2.158\alpha$$

$$x_3 = -0.3810 - 2.287\alpha$$

Performing an exact line search along the gradient gives:

$$y = 2(0.1584 + 0.6336\alpha)^2 + (-1.079 - 2.158\alpha)^2 + 3(-0.3810 - 2.287\alpha)^2$$

Setting $dy/d\alpha = 0$ and solving gives $\alpha^* = -0.2432$. Computing $\mathbf{x}_3$ gives $(0.0043, -0.5543, 0.1750)^T$.

Accelerating from $\mathbf{x}_0$ through $\mathbf{x}_3$ to locate $\mathbf{x}_4$ gives:

$$\mathbf{x} = \mathbf{x}_0 + \alpha(\mathbf{x}_3 - \mathbf{x}_0)$$

or

$$x_1 = 2 - 1.996\alpha$$

$$x_2 = -2 + 1.446\alpha$$

$$x_3 = 1 - 0.8250\alpha$$

Performing a search along the acceleration line gives:

$$y = 2(2 - 1.996\alpha)^2 + (-2 + 1.446\alpha)^2 + 3(1 - 0.8250\alpha)^2$$

Setting $dy/d\alpha = 0$ and solving gives $\alpha^* = 1.1034$. Computing $\mathbf{x}_4$ gives $(-0.2021, -0.4048, 0.0897)^T$.

The procedure is continued with a gradient search from $\mathbf{x}_4$ to $\mathbf{x}_5$ and an acceleration step from $\mathbf{x}_2$ through $\mathbf{x}_5$ to $\mathbf{x}_6$, the optimum. The following tabulates the results of these calculations and the previous ones.

		x_1	x_2	x_3	parameter of the gradient or acceleration line
Start	$\mathbf{x}_0$	2	-2	1	
Gradient					-0.2302
	$\mathbf{x}_2$	0.1584	-1.079	-0.3810	
Gradient					-0.2432
	$\mathbf{x}_3$	0.0043	-0.5543	0.1750	
Accelerate					1.1034
	$\mathbf{x}_4$	-0.2021	-0.4048	0.0897	
Gradient					-0.2822
	$\mathbf{x}_5$	0.0260	-0.1764	-0.0622	
Accelerate					1.1915
Optimum	$\mathbf{x}_6$	0.0001	0.0000	-0.0001	

The parameter of the gradient line is negative, showing that the procedure is moving in the direction of steep descent. The parameter of the acceleration line is greater than one showing the new point lies beyond the last point.

This procedure has been used successfully on numerous problems. However, it has been referred to as a "rich man's optimizer" by Wilde (10). The method tends to oscillate on problems with sharp curving ridges, and numerical computation of the gradient requires more computer time and storage than some other methods. The two equations used, the gradient and acceleration lines, are simple and easy to program; and the method will find better values in each step toward the optimum.

For those interested in a detailed discussion of conjugate gradient and direction methods, the books by Fletcher (4), Gill, et al. (6), Avriel (9), Himmelblau (8), Reklaitis et al. (15) and Wilde and Beightler (12) are recommended. Now, we will examine another class of methods that rely on logical algorithms to move rapidly from the starting point to one near an optimum.

Logical Methods: These procedures use algorithms based on logical concepts to find a sequence of improved values of the economic model leading to an optimum. They begin with local exploration, and then attempt to accelerate in the direction of success. Then if a failure occurs in that direction, the method repeats local exploration to find another direction of improved values of the economic model. If this fails, the algorithm's logic may then try other strategies including a quadratic fit of the economic model (end game) to look for better values. Typically, these

procedures do not require derivative measurements, and the algorithm compares the computed values of the economic model. Thus, they are sometimes called *function comparison methods*.

Two of the better-known methods are pattern search (12) and the polytope or simplicial method (6). Both have been used successfully on a number of problems. Pattern search is probably the more widely used of the two procedures, and it will be discussed in more detail. The polytope method performs local explorations at the vertices of an n-dimensional generalization of an equilateral triangle and can employ an acceleration step based on these results. The details of this method and extensions are given by Gill, et al. (6).

The logical algorithm of pattern search is illustrated in Figure 5-3, and it begins with short excursions from the starting point to establish a pattern of improved values of the economic model. Based on these function comparisons, it accelerates in the direction established from the local explorations. If successful, the acceleration is continued. Then when a failure is encountered, i.e. a value of the economic model is less than the previous one, the pattern is said to be destroyed; and local explorations are performed to establish a new pattern of improved values of the economic model. Again, acceleration is performed in the new direction until a failure is encountered. The procedure continues in this fashion until an apparent optimum is reached. Then the step size of the local exploration is reduced, attempting to find another direction of improvement in the economic model.

If this is successful, the procedure continues until another optimum is found. Reducing the step size is repeated; and if this is unsuccessful in finding a new direction, the current point is declared a local optimum. However, a quadratic fit at the point is needed to confirm that it is an optimum rather than a saddle point.

The algorithm has two parts. One is the local exploration procedure, and the other is the acceleration step. The local explorations are performed about a base point by perturbing one variable at a time. Each time a variable is perturbed and a better value of the economic model is found, this point is used when the next variable is changed rather than returning to the original point. These are called *temporary heads* and the first one $\mathbf{t}_{11}$ is computed by the following expression.

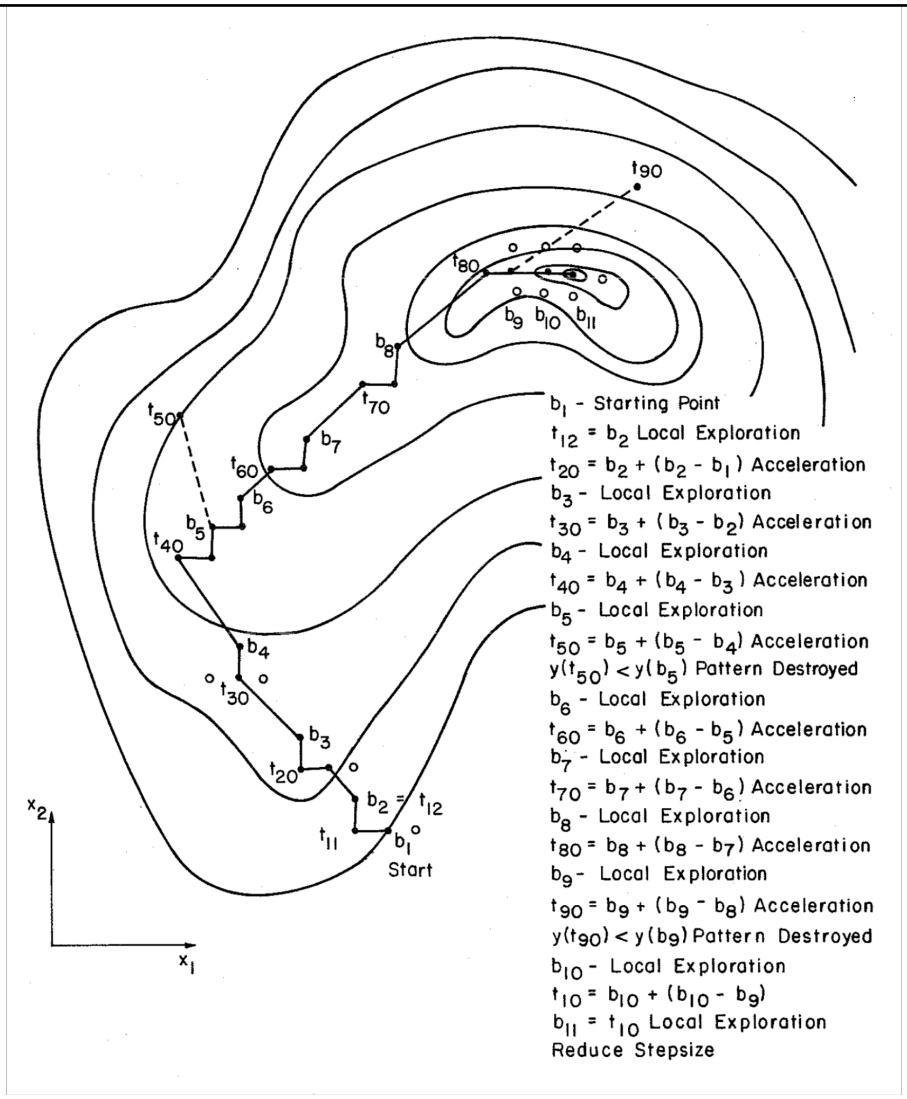


Figure 5-3 Illustration of Pattern Search

$$\mathbf{t}_{11} = \begin{cases} \mathbf{b}_1 + \boldsymbol{\delta}_1 & \text{if } y(\mathbf{b}_1 + \boldsymbol{\delta}_1) > y(\mathbf{b}) \\ \mathbf{b}_1 - \boldsymbol{\delta}_1 & \text{if } y(\mathbf{b}_1 - \boldsymbol{\delta}_1) > y(\mathbf{b}) \\ \mathbf{b}_1 & \text{if } y(\mathbf{b}) > \max\left[y(\mathbf{b}_1 + \boldsymbol{\delta}_1),\ y(\mathbf{b}_1 - \boldsymbol{\delta}_1)\right] \end{cases} \tag{5-27}$$

where $\mathbf{b}_1$ is the starting point, $\boldsymbol{\delta}_1^T = (\delta_1, 0, \ldots 0)$, and the first subscript on t_{11} refers to the pattern number and the second subscript refers to the coordinate axis of the variable being perturbed. For coordinate axis x_2 the perturbations are conducted around point $\mathbf{t}_{11}$ to locate point $\mathbf{t}_{12}$, and equation corresponding to Equation 6-27 above for the coordinate axis x_j is:

$$\mathbf{t}_{1j} = \begin{cases} \mathbf{t}_{1,j-1} + \boldsymbol{\delta}_1 & \text{if } y(\mathbf{t}_{1,j-1} + \boldsymbol{\delta}_1) > y(\mathbf{t}_{1,j-1}) \\ \mathbf{t}_{1,j-1} - \boldsymbol{\delta}_1 & \text{if } y(\mathbf{t}_{1,j-1} - \boldsymbol{\delta}_1) > y(\mathbf{t}_{1,j-1}) \\ \mathbf{t}_{1,j-1} & \text{if } y(\mathbf{t}_{1,j-1}) > \max\left[y(\mathbf{t}_{1,j-1} + \boldsymbol{\delta}_1), y(\mathbf{t}_{1,j-1} - \boldsymbol{\delta}_1)\right] \end{cases} \quad (5\text{-}28)$$

When these perturbations and evaluations are performed for each of the coordinate axes, a final point $\mathbf{t}_{1,n}$ is located. This point is designated $\mathbf{b}_2$, and an acceleration move is made in the direction established by the local exploration. This is given by the following equation and locates point $\mathbf{t}_{20}$.

$$\mathbf{t}_{20} = \mathbf{b}_1 + 2(\mathbf{b}_2 - \mathbf{b}_1) = \mathbf{b}_2 + (\mathbf{b}_2 - \mathbf{b}_1) \quad (5\text{-}29)$$

Now, point $\mathbf{t}_{20}$ is used as the starting point for local exploration following the same procedure using Equations 5-27 and 5-28 to locate point $\mathbf{b}_3$. Then the acceleration step is repeated using the same form of Equation 6-27 to locate $\mathbf{t}_{30}$.

$$\mathbf{t}_{30} = \mathbf{b}_2 + 2(\mathbf{b}_3 - \mathbf{b}_2) = \mathbf{b}_3 + (\mathbf{b}_3 - \mathbf{b}_2) \quad (5\text{-}30)$$

The search grows with repeated success.

At this point the two parts of the algorithm have been described in a general form. The local exploration step and the acceleration step can be readily implemented in a computer program, and one is given by Kuester and Mize (16). In addition, the following example illustrates the method on the contour diagram of a function of two independent variables shown in Figure 5-3. It shows the local exploration, acceleration, pattern destroyed and reestablished, and location of the optimum.

Example 5-7

Locate the maximum of the function shown in Figure 5-3 using pattern search starting at the points indicated as $\mathbf{b}_1$.

To begin, local explorations are performed by moving in the positive coordinate axis direction first (open circles indicate failures; and solid circle indicate successes). On the x_1 axis the largest of $y(x_1, x_2)$ is at $\mathbf{t}_{11}$. Then perturbing on the x_2 axis locates the largest value of y at $\mathbf{t}_{12} = \mathbf{b}_2$. Effort is not wasted by evaluating y in the negative direction on the x_2 axis.

Next, an acceleration step is performed using Equation 5-27 to locate point $\mathbf{t}_{20}$. Then local exploration determines point $\mathbf{b}_3$, and acceleration step using Equation 5-28 locates point $\mathbf{t}_{30}$. Local exploration locates point $\mathbf{b}_4$, and the acceleration step increases and changes directions as a result of the outcomes from the local exploration at $\mathbf{t}_{30}$ to reach point $\mathbf{t}_{40}$. Local exploration determines point $\mathbf{b}_5$, and acceleration gives point $\mathbf{t}_{50}$. However, $y(\mathbf{t}_{50}) < y(\mathbf{b}_5)$; and the pattern is said to be destroyed.

Local exploration is repeated, and $\mathbf{b}_6$ is located. This sequence of local explorations is repeated determining points: $\mathbf{t}_{60}$, $\mathbf{b}_7$, $\mathbf{t}_{70}$, $\mathbf{b}_8$, $\mathbf{t}_{80}$, $\mathbf{b}_9$, and $\mathbf{t}_{90}$. However, $y(\mathbf{t}_{90}) < y(\mathbf{b}_9)$ and the pattern is

destroyed. Local exploration is repeated to locate $\mathbf{b}_{10}$, and acceleration is to $\mathbf{t}_{10,0}$. However, local exploration around $\mathbf{t}_{10,0}$ shows that this point has the largest value of y and $\mathbf{t}_{10,0} = \mathbf{b}_{11}$. Then the procedure would reduce the step-size to attempt to find a direction of improvement.

Although this is not shown in Figure 5-3, the outcome would be that $y(\mathbf{b}_{11})$ is still the largest value. Point $\mathbf{b}_{11}$ would be declared a local maximum, and a quadratic fit to the function could be performed to confirm the maximum. The pattern search steps are summarized on Figure 5-3.

Pattern search has been used successfully on a number of types of problems, and it has been found to be most effective on problems with a relatively small number of independent variables e.g. ten or fewer. It has the advantage of adjusting to the terrain of a function and will follow a curving ridge. However, it can be confounded by resolution ridges (12), and a quadratic fit is appropriate to avoid this weakness.

There are a number of other methods based on logical algorithms. These are discussed in some detail in the texts by Himmelblau (8), Gill, Murray and Wright (6), and Reklaitis et al. (15). However, none of those methods are superior to the ones discussed here. Now, we will turn our attention to methods used for constrained multivariable search problems and see that the DFP and BFGS procedures are an integral part of some of these methods.

Constrained Multivariable Search Methods

There are essentially six types of procedures to solve constrained nonlinear optimization problems. The three considered most successful are successive linear programming, successive quadratic programming and the generalized reduced-gradient method. The other three have not proved as useful, especially on problems with a large number of variables (more than 20). These are penalty and barrier function methods, augmented Lagrange functions and the methods of feasible directions (or projections) that are sometimes called *methods of restricted movement*. Of these methods only successive linear programming does not require an unconstrained single or multivariable search algorithm. Also, penalty and augmented function methods have been used with successive quadratic programming. Each of these methods will be discussed in the order that they were mentioned. This will be followed by a review of studies that have evaluated the performance of the various methods.

Successive Linear Programming: This procedure was called the *method of approximate programming* (MAP) by Griffith and Stewart (18) of Shell Oil Company who originally proposed and tested the procedure on petroleum refinery optimization. As the name implies, the method uses linear programming as a search technique. A starting point is selected, and the nonlinear economic model and constraints are linearized about this point to give a linear problem that can be solved by the Simplex Method or its extensions. The point from the linear programming solution can be used as a new point to linearize the nonlinear problem, and this can be continued until a stopping criterion is met. As shown by Reklaitis et al. (15), this procedure works without safeguards for functions that are mildly nonlinear. However, it is necessary to bound the steps taken in the iterations to ensure that: the economic model improves, the values of the independent variables remain in the feasible region and the procedure converges to the optimum. These

safeguards are bounds on the independent variables specified in advance of solving the linear programming problem. The net result is that the bounds are additional constraint equations. If the bounds are set too small, the procedure will move slowly toward the optimum. If they are set too large, infeasible solutions will be generated. Consequently, logic is incorporated into computer programs to expand the bounds when they hamper rapid progress and shrink them so that the procedure may converge to a stationary point solution (1).

For successive linear programming, the general nonlinear optimization problem can be written as:

$$\begin{aligned} &\text{optimize:} && y(\mathbf{x}) && \text{(5-31)} \\ &\text{subject to:} && f_i(\mathbf{x}) \le b_i && \text{for } i = 1, 2, ..., m \\ & && u_j \ge x_j \ge l_j && \text{for } j = 1, 2, .., n \end{aligned}$$

where upper and lower limits are shown specifically on the independent variables.

Now the economic model $y(\mathbf{x})$ and the constraints $f_i(\mathbf{x})$ can be linearized around a feasible point $\mathbf{x}_k$ to give:

$$\begin{aligned} &\text{optimize: } \sum_{j=1}^{n} c_j \Delta x_j = y - y(x_k) \\ &\text{subject to: } \sum_{j=1}^{n} a_{ij} \Delta x_j \le b_i - f_i(x_j) \quad \text{for } i = 1,2,\ldots,m \\ &\qquad u_i - x_{jk} \ge \Delta x_j \ge l_j - x_{jk} \quad \text{for } j = 1,2,\ldots,n \end{aligned} \qquad \text{(5-32)}$$

$$\Delta x_j = x_j - x_{jk} \qquad c_j = \frac{\partial y(x_k)}{\partial x_j} \qquad a_{ij} = \frac{\partial f_i(x_k)}{\partial x_j}$$

The problem is in a linear programming format in the form of Equation 5-32. However, the values of Δx_j can take on either positive or negative values depending on the location of the optimum. Negative values for Δx_j are not acceptable with the Simplex Algorithm so a change of variables was made by Griffith and Stewart (18) as follows.

$$\Delta x_j = \Delta x_j^+ - \Delta x_j^- \qquad \text{(5-33)}$$

where

$$\Delta x_j^+ = \begin{cases} \Delta x_j & \text{if } \Delta x_j \ge 0 \\ 0 & \text{if } \Delta x_j < 0 \end{cases}$$

$$\Delta x_j^- = \begin{cases} -\Delta x_j & \text{if } \Delta x_j \le 0 \\ 0 & \text{if } \Delta x_j > 0 \end{cases}$$

Substituting Equation 5-33 into Equation 5-32, now the linear programming problem has the form:

$$\begin{aligned}
&\text{optimize: } \sum_{j=1}^{n} \boldsymbol{c}_j \Delta \boldsymbol{x}_j^{+} - \sum_{j=1}^{n} \boldsymbol{c}_j \Delta \boldsymbol{x}_j^{-} = \boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x}_k) \\
&\text{subject to: } \sum_{j=1}^{n} \boldsymbol{a}_{ij} \Delta \boldsymbol{x}_j^{+} - \sum_{j=1}^{n} \boldsymbol{a}_{ij} \Delta \boldsymbol{x}_j^{-} \le \boldsymbol{b}_i - \boldsymbol{f}_i(\boldsymbol{x}_j) \quad \text{for } \boldsymbol{i} = 1,2,\ldots,\boldsymbol{m} \qquad (5\text{-}34) \\
&\qquad \Delta \boldsymbol{x}_j^{+} - \Delta \boldsymbol{x}_j^{-} \le (\boldsymbol{u}_i - \boldsymbol{x}_{jk}) \qquad \text{for } \boldsymbol{j} = 1,2,\ldots,\boldsymbol{n} \\
&\qquad -\Delta \boldsymbol{x}_j^{+} + \Delta \boldsymbol{x}_j^{-} \le (\boldsymbol{x}_{jk} - \boldsymbol{l}_j)
\end{aligned}$$

The bounds on the upper and lower limits on the variables are specified by (u_j - x_{jk}) and (x_{jk} - l_j) in Equation 5-34. The inequality Δx_j^{+} - Δx_j^{-} $\geq$ (l_j - x_{jk}) is written as shown above to have a positive right hand side of these constraint equations as required by the Simplex Method.

The value of the next point for linearizing is given by $x_{jk+1} = x_{jk} + \Delta x_j^{+} - \Delta x_j^{-}$. The procedure is started by specifying a starting point $\mathbf{x}_0(k=0)$.

The above equations are now a linear programming problem where the independent variables are Δx_j^{+} and Δx_j^{-}. The value of the bound u_j and l_j may affect the rate of convergence of the algorithm. The use of bounds is illustrated in the following example given by Griffith and Stewart (18).

Example 5-8 (18)

Locate the maximum of the following constrained nonlinear optimization problem by the method of successive linear programming starting at $\mathbf{x}_0$ (1, 1), and using the bounds (u_j - x_{jk}) = (x_{jk} - l_j) = 1.

$$\begin{aligned}
&\text{maximize:} \quad y = 2x_1 + x_2 \\
&\text{subject to:} \quad x_1^2 + x_2^2 \le 25 \\
&\qquad\qquad\quad x_1^2 - x_2^2 \le 7
\end{aligned}$$

The two constraint equations are shown in Figure 5-4 where they intersect at the maximum of the economic model, point $\mathbf{x}^*(4, 3)$.

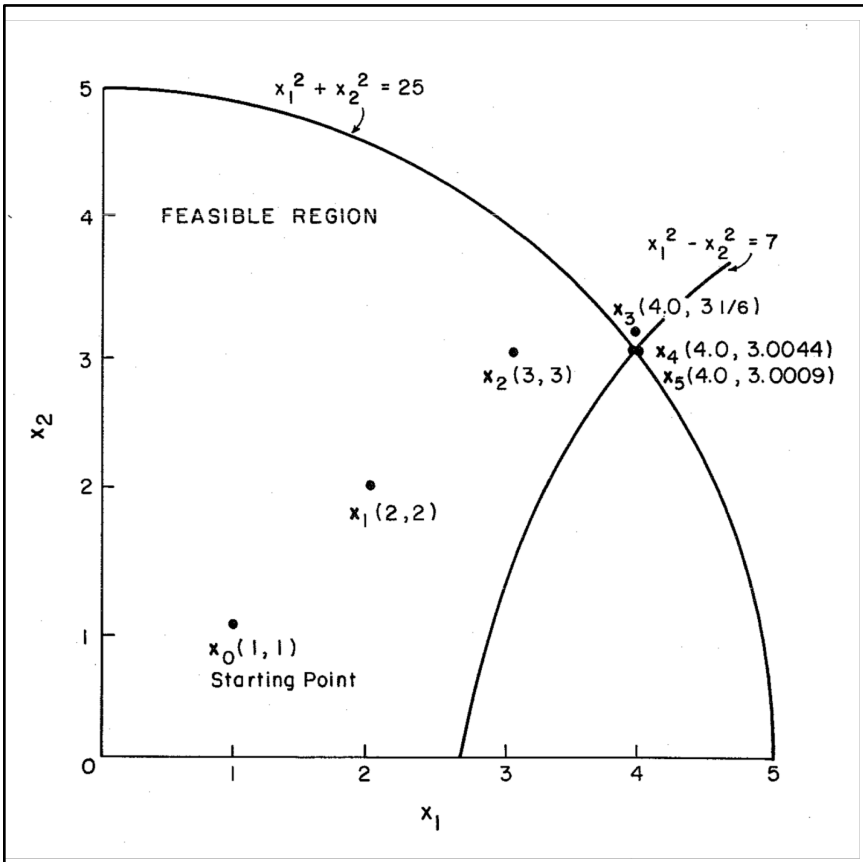


Figure 5-4 Diagram Showing the Successive Linear Programming Solution of Example 5-8 after Griffith and Stewart (18)

For this problem the successive linear programming approximation is obtained using Equation 5-34.

maximize: $2\Delta x_1^+ + \Delta x_2^+ - 2\Delta x_1^- - \Delta x_2^- = y - (2x_{1k} + x_{2k})$

subject to:

$$\begin{aligned}
2x_{1k}\,\Delta x_1^+ &+ 2x_{2k}\,\Delta x_2^+ - 2\,x_{1k}\,\Delta x_1^- - 2x_{2k}\,\Delta x_2^- &&\le 25 - [x_{1k}^2 + x_{2k}^2] \\
2x_{1k}\,\Delta x_1^+ &- 2x_{2k}\,\Delta x_2^+ - 2\,x_{1k}\,\Delta x_1^- + 2x_{2k}\,\Delta x_2^- &&\le 7 - [x_{1k}^2 - x_{2k}^2] \\
\Delta x_1^+ &- \Delta x_1^- &&\le 1 \\
&\Delta x_2^+ - \Delta x_2^- &&\le 1 \\
-\Delta x_1^+ &+ \Delta x_1^- &&\le 1 \\
&-\Delta x_2^+ + \Delta x_2^- &&\le 1
\end{aligned}$$

There are four variables in the above equations $\Delta x_1^+, \Delta x_1^-, \Delta x_2^+$, and Δx_2^-.
Starting at point $\mathbf{x}_0$ (1, 1) the above equations become:

$$\begin{aligned}
\text{maximize: } 2\Delta x_1^+ &+ \Delta x_2^+ - 2\Delta x_1^- - \Delta x_2^- &&= y - 3 \\
\\
\text{subject to: } 2\Delta x_1^+ &+ 2\Delta x_2^+ - 2\Delta x_1^- - 2\Delta x_2^- &&\le 23 \\
2\Delta x_1^+ &- 2\Delta x_2^+ - 2\Delta x_1^- + 2\Delta x_2^- &&\le 7
\end{aligned}$$

$$
\begin{array}{rrrrl}
\Delta x_1^+ & & - \Delta x_1^- & & \le 1 \\
 & \Delta x_2^+ & & - \Delta x_2^- & \le 1 \\
-\Delta x_1^+ & & + \Delta x_1^- & & \le 1 \\
 & -\Delta x_2^+ & & + \Delta x_2^- & \le 1
\end{array}
$$

Solving by the Simplex Method gives:

$$\Delta x_1^+ = 1 \qquad \Delta x_1^- = 0 \qquad \Delta x_2^+ = 1 \qquad \Delta x_2^- = 0$$

$\mathbf{x}_1$ is the computed as follows:

$$x_{1,1} = x_{1,0} + \Delta x_1^+ - \Delta x_1^- = 1 + 1 - 0 = 2$$

$$x_{2,1} = x_{2,0} + \Delta x_2^+ - \Delta x_2^- = 1 + 1 - 0 = 2$$

and

$$\mathbf{x}_1\,(2, 2) \qquad y(\mathbf{x}_1) = 6$$

Linearizing around $\mathbf{x}_2\,(2, 2)$ gives:

$$
\begin{array}{rrrrrl}
\text{Maximize: } & 2\Delta x_1^+ & + \Delta x_2^+ & - 2\Delta x_1^- & - \Delta x_2^- & = y - 6 \\
 & & & & & \\
\text{Subject to: } & 4\Delta x_1^+ & + 4\Delta x_2^+ & - 4\Delta x_1^- & - 4\Delta x_2^- & \le 17 \\
 & 4\Delta x_1^+ & - 4\Delta x_2^+ & - 4\Delta x_1^- & + 4\Delta x_2^- & \le 7 \\
 & \Delta x_1^+ & & - \Delta x_1^- & & \le 1 \\
 & & \Delta x_2^+ & & - \Delta x_2^- & \le 1 \\
 & -\Delta x_1^+ & & + \Delta x_1^- & & \le 1 \\
 & & -\Delta x_2^+ & & + \Delta x_2^- & \le 1
\end{array}
$$

Solving by the Simplex Method gives:

$$\Delta x_1^+ = 1 \qquad \Delta x_1^- = 0 \qquad \Delta x_2^+ = 1 \qquad \Delta x_2^- = 0$$

$\mathbf{x}_2$ is the computed as follows:

$$x_{1,2} = x_{1,1} + \Delta x_1^+ - \Delta x_1^- = 2 + 1 - 0 = 3$$

$$x_{2,2} = x_{1,2} + \Delta x_2^+ - \Delta x_2^- = 2 + 1 - 0 = 3$$

and $\mathbf{x}_2(3, 3) \qquad y(\mathbf{x}_2) = 9$

Note that in Figure 5-4, the movement is controlled by the step size to this point.

Linearizing around $\mathbf{x}_2$ (3, 3) gives:

$$\begin{array}{lrrrrl}
\text{maximize: } & 2\Delta x_1^+ & + \Delta x_2^+ & - 2\Delta x_1^- & - \Delta x_2^- & = y - 9 \\
\\
\text{subject to: } & 6\Delta x_1^+ & + 6\Delta x_2^+ & - 6\Delta x_1^- & - 6\Delta x_2^- & \leq 7 \\
& 6\Delta x_1^+ & - 6\Delta x_2^+ & - 6\Delta x_1^- & + 6\Delta x_2^- & \leq 7 \\
& \Delta x_1^+ & & - \Delta x_1^- & & \leq 1 \\
& & \Delta x_2^+ & & - \Delta x_2^- & \leq 1 \\
& -\Delta x_1^+ & & + \Delta x_1^- & & \leq 1 \\
& & -\Delta x_2^+ & & + \Delta x_2^- & \leq 1
\end{array}$$

Solving by the Simplex Method gives:

$$\Delta x_1^+ = 1 \qquad \Delta x_1^- = 0 \qquad \Delta x_2^+ = 1/6 \qquad \Delta x_2^- = 0$$

$\mathbf{x}_3$ is the computed as follows:

$$x_{1,3} = x_{1,2} + \Delta x_1^+ - \Delta x_1^- = 3 + 1 - 0 = 4$$

$$x_{2,3} = x_{2,2} + \Delta x_2^+ - \Delta x_2^- = 3 + 1/6 - 0 = 3\ 1/6$$

and

$$\mathbf{x}_3(4,\ 3\ 1/6) \qquad y(\mathbf{x}_3) = 11\ 1/6$$

Note that in Figure 5-4, the movement is controlled by one of the constraint equations.

Point $\mathbf{x}_3$ is slightly infeasible by 1/6 on the x_2 axis. Deciding to continue the search at this infeasible point is called following an *infeasible path strategy*. The other option is to return to point $\mathbf{x_2}$ and reduce the step size by one-half, for example. The right hand side of the last four constraint equations would be changed from 1.0 to 0.5. If the optimization program uses the infeasible path strategy, then checks are built-in to prevent increasing infeasible points.

Linearizing around $\mathbf{x}_3$ (4, 3 1/6) gives:

$$\begin{array}{lrrrrl}
\text{maximize: } & 2\Delta x_1^+ & + \Delta x_2^+ & - 2\Delta x_1^- & - \Delta x_2^- & = y - 11\ 1/6 \\
\\
\text{subject to: } & 8\Delta x_1^+ & + (19/3)\Delta x_2^+ & - 8\Delta x_1^- & - (19/3)\Delta x_2^- & \leq -37/36 \\
& 8\Delta x_1^+ & - (19/3)\Delta x_2^+ & - 8\Delta x_1^- & + (19/3)\Delta x_2^- & \leq 37/36 \\
& \Delta x_1^+ & & - \Delta x_1^- & & \leq 1 \\
& & \Delta x_2^+ & & - \Delta x_2^- & \leq 1 \\
& -\Delta x_1^+ & & + \Delta x_1^- & & \leq 1 \\
& & - \Delta x_2^+ & & + \Delta x_2^- & \leq 1
\end{array}$$

Solving by the Simplex Method gives:

$$\Delta x_1^+ = 0.0 \qquad \Delta x_1^- = 0.0 \qquad \Delta x_2^+ = 0.0 \qquad \Delta x_2^- = 0.1623$$

$\mathbf{x}_4$ is the computed as follows:

$$x_{1,4} = x_{1,3} + \Delta x_1^+ - \Delta x_1^- = 4.0 + 0.0 - 0.0 = 4.00$$

$$x_{2,4} = x_{2,3} + \Delta x_2^+ - \Delta x_2^- = 3\ 1/6 + 0.0 - 0.1623 = 3.0044$$

and

$$\mathbf{x}_4\,(4.0,\ 3.0044) \qquad\qquad y(\mathbf{x}_4) = 11.00$$

Note point $\mathbf{x}_4$ is less infeasible than point $\mathbf{x}_3$.

Linearizing around $\mathbf{x}_4(4.0, 3.0044)$ gives:

$$\begin{array}{rrrrrrrrl}
\text{Maximize:} & 2\Delta x_1^+ & + & \Delta x_2^+ & - & 2\Delta x_1^- & - & \Delta x_2^- & = y - 11.011 \\
\\
\text{Subject to:} & 8.0\Delta x_1^+ & + & 6.0088\Delta x_2^+ & - & 8.0\Delta x_1^- & - & 6.0088\Delta x_2^- & \leq -\ 0.0264 \\
& 8.0\Delta x_1^+ & - & 6.0088\Delta x_2^+ & - & 8.0\Delta x_1^- & + & 6.0088\Delta x_2^- & \leq\ \ 0.0264 \\
& \Delta x_1^+ & & & - & \Delta x_1^- & & & \leq 1 \\
& & & \Delta x_2^+ & & & - & \Delta x_2^- & \leq 1 \\
& -\Delta x_1^+ & & & + & \Delta x_1^- & & & \leq 1 \\
& & - & \Delta x_2^+ & & & + & \Delta x_2^- & \leq 1
\end{array}$$

Solving by the Simplex Method gives:

$$\Delta x_1^+ = 0.0 \qquad \Delta x_1^- = 0.0 \qquad \Delta x_2^+ = 0.0 \qquad \Delta x_2^- = 0.00438$$

$\mathbf{x}_5$ is the computed as follows:

$$x_{1,5} = x_{1,4} + \Delta x_1^+ - \Delta x_1^- = 4.0 + 0.0 - 0.0 = 4.00$$

$$x_{2,5} = x_{2,4} + \Delta x_2^+ - \Delta x_2^- = 3.0044 + 0.0 - 0..00438 = 3.0000$$

and

$$\mathbf{x}_5(4.0,\ 3.0000) \qquad\qquad y(\mathbf{x}_5) = 11.00$$

This is the optimal solution and is the same as given by Griffith and Stewart (18).

It should be noted that point $\mathbf{x}_3$ (4, 3 1/6) is an infeasible point and does not satisfy the first constraint equation. However, this point is sufficiently close to the optimum that the method converges to the optimum after linearizing around this point. Convergence to the optimum will not take place if bounds are not used, however.

This problem was solved without the constraints bounding the variables, i.e. omitting the last four constraint equations. Starting at point $\mathbf{x}_0(1, 1)$ the point $\mathbf{x}_1(8.5, 5.0)$ was found. Linearizing around $\mathbf{x}_1$ (8.5, 5.0) and solving by the Simplex Method gave the point $\mathbf{x}_2(0,12.23)$.

Then linearizing around $\mathbf{x}_2(0, 12.23)$ gave a set of constraint equations that had an unbounded solution. Consequently, bounds were required on this problem to ensure convergence to a solution.

Computer programs can reduce the bounds when an infeasible solution is located and resolve the problem. This was done for the problem starting at point $\mathbf{x}_2(3, 3)$ since point $\mathbf{x}_3(4, 3\ 1/6)$ was infeasible, and the bounds were reduced by one-half each time an infeasible point was obtained. Following this procedure, the next two iterations for this problem were (3.563, 3.492) and (3.595, 3.475). Further examination showed the method had difficulty following the first constraint to the optimum. As Himmelblau (8) points out, when constraints become active then successive linear programming's "progress becomes quite slow." Consequently, logic is incorporated in some programs to allow the procedure to continue from an infeasible point, as was done by Griffith and Stewart in this example.

For those interested in having a successive linear programming code, Lasdon (19) reports that the most widely used and best known one, POP (Process Optimization Procedure) is available from the SHARE library (COSMIC, Bartow Hall, University of Georgia; Athens GA 30601). Other listings of sources of optimization codes are given by Sandgren (20) and Lasdon and Waren (22).

Large linear programming codes have been used with large simulation models in an iterative fashion to approximate the nonlinearities in these models. This approach of using linear programming successively has been successful in large plants. In most cases, this procedure has been used by companies that have many man-years of effort in the development and use of a large linear programming code for plant optimization and a corresponding amount of effort in large simulations of key process units for prediction of performance and yields. An example of this is in petroleum refining where linear programming is used for refinery optimization. In addition, elaborate simulations and correlations have been developed for processes such as catalytic cracking, reforming and distillation.

As discussed in Chapter 3, the results of a linear programming optimization are as accurate as the parameters in the economic model and constraint equations, **c**, **A** and **b**. As shown in Figure 5-5 iterative procedures have been developed that use these programs together. The large simulation codes are used to compute the parameters used in the large linear programming code. Then the linear programming code is used to generate an optimal solution in terms of the independent variables, **x**, which are the process variables required by the simulation codes. This iteration procedure is continued until a stopping criterion is met. Both the linear programming code and the process simulators are very large programs, and no attempt is made to have them run at the same time. Typically, the output from the simulators is edited by a separate program to produce a data set in the form required by the linear programming code. Also, another program can be used to manipulate the output from the linear programming code into a data set for use by the simulation programs. Further descriptions of these procedures are given by Pollack and Lieder (31) for petroleum refinery optimization and by O'Neil, et al. (32) for the allocation of natural gas in large pipeline networks.

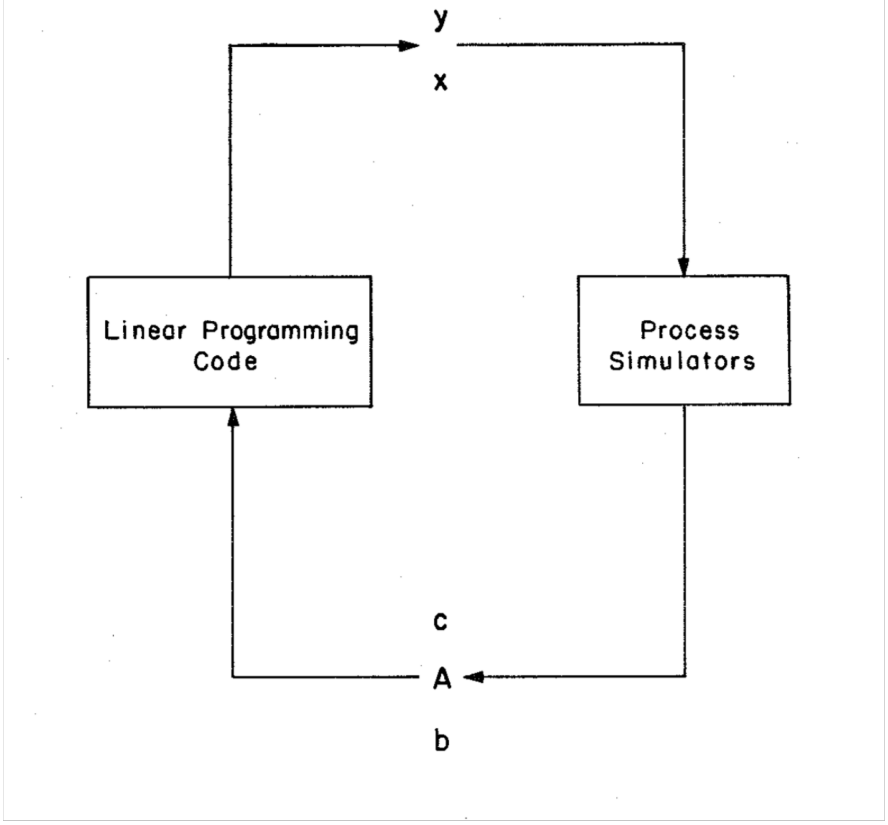


Figure 5-5 Diagram Showing the Use of Process Simulators with Linear Programming

Successive Quadratic Programming: Like successive linear programming, a quadratic programming problem is formed from the nonlinear programming problem, and it is solved iteratively until an optimum is reached. However, the iterative procedure differs from that of successive linear programming. As described by Lasdon and Waren (22), the quadratic programming solution is not accepted immediately as the next point to continue the search, but a single variable search is performed between the old and new points to have a better and feasible point.

In quadratic programming the economic model is a quadratic function, and the constraints are all linear equations. To solve this problem the Lagrange function is formed, and the Kuhn-Tucker conditions are applied to the Lagrange function (23, 24, 25) to obtain a set of linear equations. This set of linear equations can then be solved by the Simplex Method for the optimum. It turns out that artificial variables are required for part of the constraints and the slack variables can be used for the other constraints to have an initially feasible basis. Also, finding an initial basic feasible solution may be the only feasible solution (25), so the linear programming

computational effort is minimal. At this point it is important to understand the solution of a quadratic programming problem, and this procedure will be described next and illustrated with an example. Then the successive quadratic programming algorithm will be described and illustrated with an example. Also, modifications of the procedure will be discussed that reduce the computational effort in numerically evaluating the Hessian matrix that must be obtained from the nonlinear programming problem.

Theoretically, using a quadratic function to approximate the nonlinear economic model of the process can be considered superior to a linear function to represent the economic model. This is part of the motivation for using quadratic programming that can be represented by the following equations:

$$\text{maximize: } \sum_{j=1}^{n} c_j x_j - \tfrac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} q_{jk} x_j x_k$$

$$\text{subject to: } \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad \text{for } i = 1,2,\ldots,m \tag{5-35}$$

$$x_j \ge 0 \quad \text{for } j = 1,2,\ldots,n$$

where $q_{jk} = q_{kj}$ would be the second partial derivatives with respect to x_j and x_k of the nonlinear economic model. They would be computed numerically or analytically from the nonlinear problem given by Equation 5-31. Also, c_j and a_{ij} would be computed as shown by Equation 5-32 either numerically or analytically from the nonlinear problem, Equation 5-31.

The quadratic programming procedure begins by adding slack variables x_{n+i} to the linear constraint equations. It will not be necessary to use x_{n+i}^2, since the problem will be solved by linear programming, and all of the variables must be positive or zero. The Lagrange function is formed as follows:

$$\begin{aligned} L(x,\lambda) &= \sum_{j=1}^{n} c_j x_j - \tfrac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} q_{jk} x_j x_k \\ &- \sum_{i=1}^{m} \lambda_i \left(\sum_{j=1}^{n} a_{ij} x_j + x_{n+i} - b_i \right) \\ &- \sum_{j=1}^{n} \lambda_{m+j} (-x_j + x_{sj}) \end{aligned} \tag{5-36}$$

In the second term of Equation 5-36, positive Lagrange multipliers are required, so a negative sign is used on this term with the constraint equations. (See Equation 2-49.) The third term is included

to ensure the variables are positive or zero, i.e. $x_j \geq 0$ or $-x_j \leq 0$ which is written as an equality – $x_j + x_{sj} = 0$ with slack variables, x_{sj}.

Setting the first partial derivatives of the Lagrange function with respect to x_j and λ_i equal to zero give the following set of $(n + m)$ linear algebraic equations:

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{x}_j} = \boldsymbol{c}_j - \sum_{k=1}^{n} \boldsymbol{q}_{jk}\boldsymbol{x}_k - \sum_{i=1}^{m} \boldsymbol{a}_{ij}\lambda_i + \lambda_{m+j} = 0 \quad \text{for j=1,2,...,n} \tag{5-37}$$

$$\sum_{j=1}^{n} \boldsymbol{a}_{ij}\boldsymbol{x}_j + \boldsymbol{x}_{n+i} - \boldsymbol{b}_i = 0 \quad \text{for } \boldsymbol{i} = 1,2,\ldots,\boldsymbol{m} \tag{5-38}$$

Considering λ_{m+j} as a slack variable, Equation 5-37 can be written as:

$$\boldsymbol{c}_j - \sum_{k=1}^{n} \boldsymbol{q}_{jk}\boldsymbol{x}_k - \sum_{i=1}^{m} \boldsymbol{a}_{ij}\lambda_i \leq 0 \quad \text{for j=1,2,...,n} \tag{5-39}$$

The inequality form of the Kuhn - Tucker conditions, Equation 5-38, is used to account for $x_j \geq 0$. (See Hillier and Lieberman (25), and Hadley (59)).

$$\sum_{j=1}^{n} \boldsymbol{a}_{ij}\boldsymbol{x}_j + \boldsymbol{x}_{n+i} = \boldsymbol{b}_i \quad \text{for } \boldsymbol{i} = 1,2,\ldots,\boldsymbol{m} \tag{5-40}$$

Also, the complementary slackness conditions must be satisfied, i.e. product of the slack variables x_{n+i} and the Lagrange multipliers λ_i are zero.

$$\lambda_i \boldsymbol{x}_{n+i} = 0 \quad \text{for } \boldsymbol{i} = 1,2,\ldots,\boldsymbol{m} \tag{5-41}$$

If $x_{n+i} = 0$, then the constraint is active, an equality; and $\lambda_i \neq 0$. However, if $x_{n+i} \neq 0$, then the constraint is inactive, an inequality; and $\lambda_i = 0$. For more details refer to the discussion in Chapter 2.

The set of Equations 5-39 and 5-40 can be converted to constraint equations for a linear programming problem in the following way. Surplus variables are added to Equation 5-39 as s_j, and slack variables are added to Equation 6-40 as x_{n+i}. The slack variables x_{n+i} can serve as the variables for an initially feasible basis for Equations 5-40. However, artificial variables are required to have an initially feasible basis for Equation 5-39. Adding artificial variables z_j with a coefficient c_j to Equations 5-39 is a convenient way to start with an initially feasible basis with $z_j = 1$. Also, the objective function will be to minimize the sum of the artificial variables, z_j, to ensure that they will not be in the final optimal solution. As a result of these modifications, Equations 5-39 and 5-40 become the constraints in the following linear programming problem:

$$\text{minimize: } \sum_{j=1}^{n} z_j$$

$$\text{subject to: } \sum_{k=1}^{n} q_{jk}x_k + \sum_{i=1}^{m} a_{ij}\lambda_i - s_j + c_j z_j = c_j \quad \text{for j=1,2,...,n} \qquad (5\text{-}42)$$

$$\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i \quad \text{for } i = 1,2,\ldots,m$$

This is now a linear programming problem which can be solved for optimal values of **x** and **λ**, the solution of the quadratic programming problem. In addition, the solution must satisfy $\mathbf{x} \geq 0$, $\boldsymbol{\lambda} \geq 0$ and $\lambda_i x_{n+i} = 0$. Consequently, the Simplex Algorithm has to be modified to avoid having both λ_i and x_{n+i} be basic variables, i.e. nonzero, to satisfy the complimentary slackness conditions (26). This may require choosing the second, best variable to enter the basis in proceeding with the Simplex Algorithm if either λ_i or x_{n+i} are in the basis and the other one is to enter.

Franklin (23) has given uniqueness and existence theorems that prove the above procedure is the solution to the quadratic programming problem and is a recommended reference for those details. At this point the method is illustrated with an example.

Example 5-9 (25)

Using quadratic programming determine the maximum of the following function subject to the constraint given.

$$\text{maximize: } 5x_1 + x_2 - 1/2(2x_1^2 - 2x_1x_2 - 2x_2x_1 + 2x_2^2)$$

$$\text{subject to: } x_1 + x_2 \leq 2$$

The quadratic programming problem is constructed using Equation 5-42 with $c_1 = 5$, $c_2 = 1$, $q_{11} = 2$, $q_{12} = -2$, $q_{21} = -2$, $q_{22} = 2$, $a_{11} = 1$, $a_{12} = 1$ and $b_1 = 2$.

The linear programming problem from Equation 5-42 is:

$$\text{minimize: } z_1 + z_2$$

$$\begin{aligned} \text{subject to: } & 2x_1 - 2x_2 + \lambda_1 - s_1 + 5z_1 = 5 \\ & -2x_1 + 2x_2 + \lambda_1 - s_2 + z_2 = 1 \\ & x_1 + x_2 + x_3 = 2 \end{aligned}$$

Eliminating z_1 and z_2 from the objective function gives the following set of equations for the application of the Simplex Method.

$1\ 3/5\, x_1 - 1\ 3/5 x_2 - 1\ 1/5\lambda_1 + 1/5 s_1 + s_2 = C\text{-}2$	$C = 2$
$2x_1 - 2x_2 + \lambda_1 - s_1 + 5z_1 = 5$	$z_1 = 1$
$-2x_1 + 2x_2 + \lambda_1 - s_2 + z_2 = 1$	$z_2 = 1$
$x_1 + x_2 + x_3 = 2$	$x_3 = 2$

x_2 enters the basis, z_2 leaves the basis

$0x_1 - 2/5\lambda_1 + 1/5\, s_1 + 1/5\, s_2 + 4/3 z_2 = C\text{-}1\ 1/5$	$C = 1\ 1/5$
$2\lambda_1 - s_1 - s_2 + 5z_1 + z_2 = 6$	$z_1 = 6/5$
$-x_1 + x_2 + ½\,\lambda_1 - ½\, s_2 + ½\, z_2 = 1/2$	$x_2 = ½$
$2x_1 + x_3 - ½\,\lambda_1 + ½\, s_2 - ½\, z_2 = 1\ ½$	$x_3 = 1\ ½$

λ_1 would enter the basis, and the second constraint equation would be used for algebraic manipulations to ensure a positive right-hand side of the constraint equations according to the Simplex Algorithm. However, this would have both λ_1 and x_3 in the basis (nonzero); and the complementary slackness conditions, $\lambda_1 x_3 = 0$, would not be satisfied. Consequently, another variable must be selected to enter the basis. This is usually the one with the next small coefficient and for this problem is x_1. Select x_1 to enter the basis, and x_3 leaves the basis.

$-2/5\,\lambda_1 + 1/5\, s_1 + 1/5\, s_2 + 4/5\, z_2 = C\ \text{-}1\ 1/5$	$C = 1\ 1/5$
$2\ \lambda_1 - s_1 - s_2 + 5z_1 + z_2 = 6$	$z_1 = 6/5$
$x_2 + ½ x_3 + ¼\ \lambda_1 - ¼\ s_2 + ¼\, z_2 = 1\ ¼$	$x_2 = 1\ ¼$
$x_1 + ½\, x_3 - ¼\ \lambda_1 + ¼\ s_2 - 1/4\, z_2 = ¾$	$x_1 = ¾$

λ_1 enters the basis, z_1 leaves the basis.

$z_1 + z_2 = C - 0$	$C = 0$
$\lambda_1 - 1/3\, s_1 - 1/3\, s_2 + 5/3\, z_1 + 1/3\, z_2 = 3$	$\lambda_1 = 3$
$x_2 + x_3 + 1/12\, s_1 + 1/12\, s_2 - 5/12\, z_1 - 1/6\, z_2 = 1/2$	$x_2 = 1/2$
$x_1 + x_3 - 1/12\, s_1 - 1/12\, s_2 + 5/12\, z_1 - 1/6\, z_2 = 3/2$	$x_1 = 3/2$

The minimum has been reached. All of the coefficients of the variables in the objective function are positive. Therefore, the optimal solution to this quadratic programming problem is:

$$x_1 = 3/2 \qquad x_2 = ½ \qquad \lambda_1 = 3 \qquad x_3 = 0$$

The positive Lagrange multiplier is consistent with the Kuhn-Tucker conditions for a maximum, Equation 2-48, since a negative sign was used in Equation 5-36.

Successive quadratic programming iteratively solves a nonlinear programming problem by using a quadratic approximation to the economic model and a linear approximation to the constraint equations. As the series of quadratic programming problems are solved, these intermediate solutions generate a sequence of points that must remain in the feasible region or sufficiently close to this region to converge to the optimum. The logic used with this method is to search along the line between the new and previous point to maintain a feasible or near feasible solution. Also, the computational effort in evaluating the Hessian matrix is significant, and quasi-Newton approximations have been used to reduce this effort. The following example illustrates successive quadratic programming for a simple problem. The discussion that follows describes modifications to the computational procedure to improve the efficiency of the method.

Example 5-10

Solve the following problem by successive quadratic programming starting at point $\mathbf{x}_0$ (0,0).

$$\text{minimize: } (x_1 - 1)^2 + (x_2 - 2)^2$$

$$\text{subject to: } 0.104x_1^2 - 0.75x_1 + x_2 \leq 0.85$$

$$x_1 + x_2 \leq 4.0$$

The contours of the economic model and the constraint equations are shown in Figure 5-6.

The nonlinear constraint equation is linearized about the point $\mathbf{x}_k$, and it has the following form.

$$(0.208x_{1k} - 0.75)x_1 + x_2 \leq 0.85 + 0.104x_{1k}^2$$

Placing the problem in the form of Equation 5-35, gives:

$$\text{maximize: } 2x_1 + 2x_2 - 1/2\ (2x_1^2 + 2x_2^2) - 5$$

$$\text{subject to: } (0.208x_{1k} - 0.75)x_1 + x_2 \leq 0.85 + 0.104x_{1k}^2$$

$$x_1 + x_2 \leq 4$$

The quadratic programming problem is constructed using Equation 5-42 with $c_1 = 2$, $c_2 = 4$, $q_{11} = 2$, $q_{12} = q_{21} = 0$, $q_{22} = 2$, $a_{11} = (0.208x_{1k} - 0.75)$, $a_{12} = 1$, $a_{21} = 1$, $a_{22} = 1$, $b_1 = 0.85 + 0.104x_{1k}$, $b_2 = 4$:

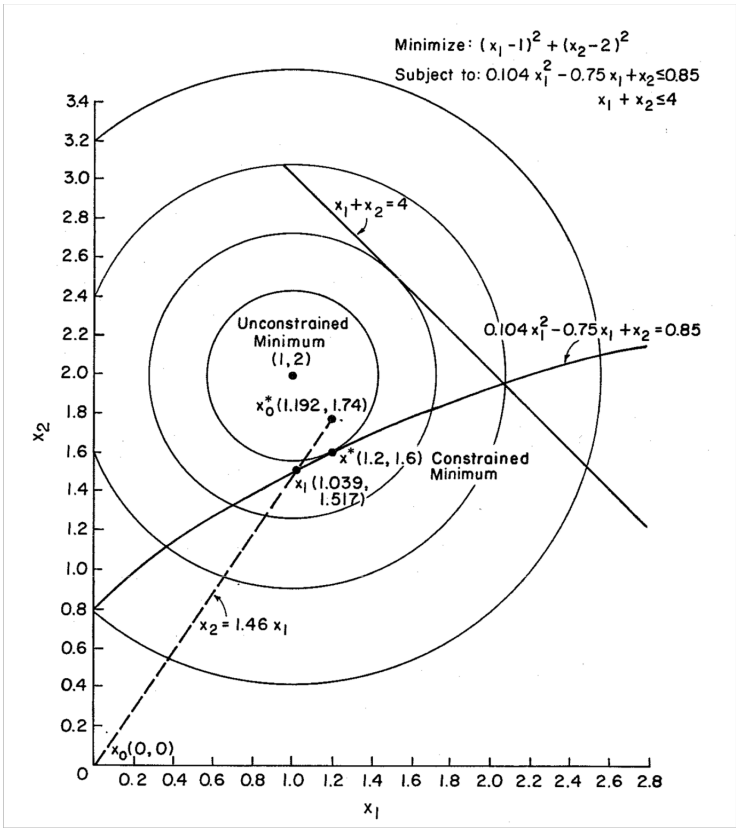


Figure 5-6 Diagram of Solution of the Multivariable Optimization Problem in Example 5-10 by Successive Quadratic Programming

minimize: $z_1 + z_2$

subject to:

$$2x_1 + (0.208x_{1k} - 0.75)\lambda_1 + \lambda_2 - s_1 + 2z_1 = 2$$

$$2x_1 + \lambda_1 + \lambda_2 - s_2 + 4z_2 = 4$$

$$(0.208x_{1k} - 0.75)x_1 + x_2 + x_3 = 0.85 + 0.104x_{1k}^2$$

$$x_1 + x_2 + x_4 = 4$$

Solving the above linear programming problem by the Simplex Method with $\mathbf{x}_0 = (0, 0)$ and ensuring that the complementary slackness conditions are met gives the following result for $\mathbf{x}_0$*.

$x_1 = 1.192 \qquad x_2 = 1.740 \qquad \lambda_1 = 0.512$

This point is shown on Figure 5-6, and it is outside the feasible region. Consequently, a search along the line between the starting point $\mathbf{x}_0$ (0,0) and $\mathbf{x}_0^*$ (1.192, 1.740) locates feasible point $\mathbf{x}_1$(1.039, 1.517) on the first constraint.

The quadratic programming problem is formulated around point $\mathbf{x}_1$ and is solved as was done above. The result is $\mathbf{x}_1^*$ (1.209, 1.608) with $\lambda_1 = 0.784$ which is infeasible but "close enough" to continue with this point becoming $\mathbf{x}_2$ on an "infeasible path."

$$x_1 = 1.209 \qquad x_2 = 1.608 \qquad \lambda_1 = 0.784$$

Repeating the procedure by solving the quadratic programming problem at $\mathbf{x}_2$ gives the value of $\mathbf{x}_2^*$ (1.199, 1.600) with $\lambda_1 = 0.800$. This point is feasible and is called $\mathbf{x}_3$.

$$x_1 = 1.199 \qquad x_2 = 1.600 \qquad \lambda_1 = 0.800.$$

This point is sufficiently close to the optimum of the problem $\mathbf{x}^*$ (1.2, 1.6) for the purposes of this illustration to say that a converged solution has been obtained.

The Wilson-Han-Powell method is an enhancement to successive quadratic programming where the Hessian matrix, $[q_{jk}]$ of Equation 5-35, is replaced by a quasi-Newton update formula such as the BFGS algorithm, Equation 5-20. Consequently, only first partial derivative information is required, and this is obtained from finite difference approximations of the Lagrange function, Equation 5-36. Also, an exact penalty function is used with the line search to adjust the step from one feasible point to the next feasible point. The theoretical basis for this algorithm is that it has a superlinear convergence rate if an exact penalty function is used with the DFP or BFGS update for the Hessian matrix of the Lagrange function, and global convergence is obtained to a Kuhn-Tucker point when minimizing an economic model that is bounded below and has convex functions for constraint equations. The details and proofs are given by Han (51,52).

The problem in Example 5-10 was solved with the Wilson-Han- Powell algorithm. The identity matrix was used for the Hessian matrix at the starting point $\mathbf{x}_0$ (0, 0). The subsequent steps in the solution were $\mathbf{x}_1$ (1.3803, 1.6871), $\mathbf{x}_2$ (1.203, 1.6038), and $\mathbf{x}_3$ (1.1988, 1.603), which was sufficiently close to the optimum to stop. Generally, less computational effort is required with the Wilson-Han-Powell algorithm since second order partial derivatives do not have to be evaluated.

The Exxon quadratic programming code (1) uses the Wilson-Han-Powell algorithm described above, and they have added refinements to minimize the computational effort in evaluating the second partial derivatives of the Hessian matrix. This typical large quadratic programming code is described as having the following steps of basic logic. An initial starting point is selected, and the linearized constraints are constructed numerically. Then the matrix of second partial derivatives, the Hessian matrix, is evaluated either numerically or a DFP (Davidon, Fletcher, Powell) approximation can be used. The quadratic programming problem is solved generating a new optimal point. Using this new point and the old point, a single variable search is conducted for an improved, feasible solution to the nonlinear problem. This is followed by changes in step and function values, feasibility checks and termination tests using the Kuhn-Tucker

conditions. Some options included in the program include using analytical derivatives when furnished, inputting the Hessian matrix by the user or having it be a specified multiple of the identity matrix with up-dating by the DFP algorithm, and having the user specify whether or not intermediate solutions are required to be feasible.

In closing this section, it should be mentioned that the Wilson- Han-Powell (WHP) method has been used successfully on computer-aided process design problems, as described by Jirapongphan, et al. (42), Vanderplaats (24) and Biegler and Cuthrell (53). In some applications, the constraint equations were not converged for each step taken by the optimization algorithm, but an infeasible trajectory was followed where the constraints were not satisfied until the optimum was reached. In the line search to adjust the step from one point to the next, an exact penalty function was used. A step length parameter was employed with the penalty function to force convergence from poor starting conditions. The size of the quadratic programming problem was reduced by substituting the linearized equality constraint equations into the quadratic economic model leaving only the inequalities as constraints. The result can be a significant reduction in the number of the independent variables for highly constrained problems.

The successive quadratic programming method has been shown to be one of the three better procedures. Now, the equally successful procedure called the *generalized reduced gradient method,* is described.

Generalized Reduced Gradient Method: This procedure is one of a class of techniques called *reduced-gradient* or *gradient projection methods* that are based on extending methods for linear constraints to apply to nonlinear constraints (6). They adjust the variables, so the active constraints continue to be satisfied as the procedure moves from one point to another. The ideas for these algorithms were devised by Wilde and Beightler (12) using the name of *constrained derivatives*, by Wolfe (29) using the name of the *reduced-gradient method* and extension by Abadie and Carpenter (30) using the name *generalized reduced gradient*. According to Avriel (9) if the economic model and constraints are linear this procedure is the Simplex Method of linear programming, and if no constraints are present it is gradient search.

The development of the procedure begins with the nonlinear optimization problem written with equality constraints. The necessary slack and surplus variables have been added as x_s or x_s^2 to any inequality constraints, and the problem is:

$$\text{optimize:} \quad y(\mathbf{x}) \tag{5-43}$$

$$\text{subject to:} \quad f_i(\mathbf{x}) = 0 \qquad \text{for } i = 1, 2, \ldots, m$$

Again, there are m constraint equations and n independent variables with $n > m$. Also, although not specifically written above, the variables can have upper and lower limits; and the procedure as described here will ensure that all variables are positive or zero.

The idea of generalized reduced gradient is to convert the constrained problem into an unconstrained one by using direct substitution. If direct substitution were possible it would reduce

the number of independent variables to $(n - m)$ and eliminate the constraint equations. However, with nonlinear constraint equations, it is not feasible to solve the m constraint equations for m of the independent variables in terms of the remaining $(n - m)$ variables and then to substitute to these equations into the economic model. Therefore, the procedures of constrained variation and Lagrange multipliers in the classical theory of maxima and minima are required. There, the economic model and constraint equations were expanded in a Taylor series, and only the first order terms were retained. Then with these linear equations, the constraint equations could be used to reduce the number of independent variables. This led to the Jacobian determinants of the method of constrained variation and the definition of the Lagrange multiplier being a ratio of partial derivatives as was shown in Chapter 2.

The development of the generalized reduced gradient method follows that of constrained variation. The case of two independent variables and one constraint equation will be used to demonstrate the concept, and then the general case will be described. Consider the following problem:

$$\begin{aligned} &\text{optimize:} && y(x_1, x_2) \\ &\text{subject to:} && f(x_1, x_2) = 0 \end{aligned} \tag{5-44}$$

Expanding the above in a Taylor series about a feasible point $\mathbf{x}_k\,(x_{1k}, x_{2k})$ gives:

$$\begin{aligned} \boldsymbol{y}(\boldsymbol{x}) &= \boldsymbol{y}(\boldsymbol{x}_k) + \frac{\partial \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_1}(\boldsymbol{x}_1 - \boldsymbol{x}_{1k}) + \frac{\partial \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_2}(\boldsymbol{x}_2 - \boldsymbol{x}_{2k}) \\ 0 &= \boldsymbol{f}(\boldsymbol{x}_k) + \frac{\partial \boldsymbol{f}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_1}(\boldsymbol{x}_1 - \boldsymbol{x}_{1k}) + \frac{\partial \boldsymbol{f}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_2}(\boldsymbol{x}_2 - \boldsymbol{x}_{2k}) \end{aligned} \tag{5-45 a and b}$$

Substituting Equation 5-44b into Equation 5-44a to eliminate x_2 gives, after some rearrangement:

$$\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{x}_k) - \frac{\partial \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_2}\left(\frac{\partial \boldsymbol{f}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_2}\right)^{-1} \boldsymbol{f}(\boldsymbol{x}_k) + \left(\frac{\partial \boldsymbol{f}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_2}\right)^{-1}\left[\frac{\partial \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_1}\frac{\partial \boldsymbol{f}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_2} - \frac{\partial \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_2}\frac{\partial \boldsymbol{f}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_1}\right](\boldsymbol{x}_1 - \boldsymbol{x}_{1k}) \tag{5-46}$$

In Equation 5-46 the first two terms on the right-hand side are known constants being evaluated at point $\mathbf{x}_k$. The coefficient of $(x_1 - x_{1k})$ of the third term is a known constant, and this term gives the x_1 direction to move toward the optimum as in steep ascent. To compute the stationary point for this equation, $dy/dx_1 = 0$; and the result is the same as for constrained variation, Equation 2-18. The term in the brackets of Equation 5-45 is solved together with the constraint equation for the stationary point. However, the term in the bracket also can be viewed as giving the direction to move away from $\mathbf{x}_k$ to obtain improved values of the economic model and satisfy the constraint equation.

The generalized reduced gradient method uses the same approach as described above for two independent variables, which is to find an improved direction for the economic model and also to satisfy the constraint equations. This leads to an expression for the reduced gradient from Equation 6-43. To develop this method, the independent variables are separated into basic and nonbasic ones. There are m basic variables $\mathbf{x}_b$, and $(n - m)$ nonbasic variables $\mathbf{x}_{nb}$.

$$\boldsymbol{f}_i(\boldsymbol{x}) = \boldsymbol{f}_i(\boldsymbol{x}_b, \boldsymbol{x}_{nb}) = 0 \qquad \text{for } \boldsymbol{i} = 1,2,\ldots,\boldsymbol{m} \tag{5-47}$$

In theory the m constraint equations could be solved for the m basic variables in terms of the $(n - m)$ nonbasic variables. Indicating the solution of $\mathbf{x}_b$ in terms of $\mathbf{x}_{nb}$ from Equation 5-47 gives:

$$\boldsymbol{x}_{i,b} = \tilde{\boldsymbol{f}}(\boldsymbol{x}_{nb}) \qquad \text{for } \boldsymbol{i} = 1,2,\ldots,\boldsymbol{m} \tag{5-48}$$

The names *basic* and *nonbasic* variables are from linear programming. In linear programming the basic variables are all positive, and the nonbasic variables are all zero. However, in nonlinear programming, the nonbasic variables are used to compute the values of the basic variables and are manipulated to obtain the optimum of the economic model.

The economic model can be thought of as a function of the nonbasic variables only that is if the constraint equations, Equation 5-48, are used to eliminate the basic variables i.e.

$$\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{x}_b \boldsymbol{x}_{nb}) = \boldsymbol{y}\left[\tilde{\boldsymbol{f}}(\boldsymbol{x}_{nb}), \boldsymbol{x}_{nb}\right] = \boldsymbol{Y}(\boldsymbol{x}_{nb}) \tag{5-49}$$

Expanding Equation 5-49 in a Taylor series about $\mathbf{x}_k$ and including only the first order terms gives:

$$\sum_{j=1}^{m} \frac{\partial \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_{j,b}} \boldsymbol{dx}_{j,b} + \sum_{j=m+1}^{n} \frac{\partial \boldsymbol{y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_{j,nb}} \boldsymbol{dx}_{j,nb} = \sum_{j=m+1}^{n} \frac{\partial \boldsymbol{Y}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_{j,nb}} \boldsymbol{dx}_{j,nb} \tag{5-50}$$

In matrix notation Equation 5-50 can be written as:

$$\nabla^T \boldsymbol{Y}(\boldsymbol{x}_k) \boldsymbol{dx}_{nb} = \nabla^T \boldsymbol{y}_b(\boldsymbol{x}_k) \boldsymbol{dx}_b + \nabla^T \boldsymbol{y}_{nb}(\boldsymbol{x}_k) \boldsymbol{dx}_{nb} \tag{5-51}$$

This equation is comparable to Equation 5-45a.

A Taylor series expansion of the constraint equations, Equation 5-47, gives Equation 5-52 that can be substituted into Equation 5-51 to eliminate the basic variables and have an equation only in terms of the nonbasic variables.

$$\sum_{j=1}^{m} \frac{\partial f_i(x_k)}{\partial x_j} dx_{j,b} + \sum_{j=m+1}^{n} \frac{\partial f_i(x_k)}{\partial x_j} dx_{j,nb} = 0 \qquad \text{for } i = 1,2,\ldots,m \tag{5-52}$$

or in matrix form Equation 5-52 is:

$$\begin{bmatrix} \frac{\partial f_1(x_k)}{\partial x_1} & \cdots & \frac{\partial f_1(x_k)}{\partial x_m} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_m(x_k)}{\partial x_1} & \cdots & \frac{\partial f_m(x_k)}{\partial x_m} \end{bmatrix} \begin{bmatrix} dx_{1,b} \\ \vdots \\ dx_{m,b} \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1(x_k)}{\partial x_{m+1}} & \cdots & \frac{\partial f_1(x_k)}{\partial x_n} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_m(x_k)}{\partial x_{m+1}} & \cdots & \frac{\partial f_m(x_k)}{\partial x_n} \end{bmatrix} \begin{bmatrix} dx_{m+1,nb} \\ \vdots \\ dx_{n,nb} \end{bmatrix} = 0 \tag{5-53}$$

The following equation defines $\mathbf{B}_b$ as the matrix of the first partial derivatives of f_i associated with the basic variables, $\mathbf{x_b}$, and $\mathbf{B}_{nb}$ as the matrix associated with the non-basic variables, $\mathbf{x_{nb}}$, i.e.:

$$B_b dx_b + B_{nb} dx_{nb} = 0 \tag{5-54}$$

This is a convenient form of Equation 6-53 that can be used to eliminate $d\mathbf{x}_b$ from Equation 6-51. Solving Equation 6-54 for $d\mathbf{x}_b$ gives:

$$dx_b = -B_b^{-1} B_{nb} dx_{nb} \tag{5-55}$$

Substituting Equation 5-55 into Equation 5-51 gives:

$$\nabla^T Y(x_k) dx_{nb} = -\nabla^T y_b(x_b) B_b^{-1} B_{nb} dx_{nb} + \nabla^T y_{nb}(x_k) dx_{nb} \tag{5-56}$$

Eliminating $d\mathbf{x}_{nb}$ from Equation 5-56, the equation for the reduced gradient $\nabla^T \boldsymbol{Y}(\mathbf{x}_k)$ is obtained.

$$\nabla^T Y(x_k) = \nabla^T y_{nb}(x_k) - \nabla^T y_b(x_b) B_b^{-1} B_{nb} \tag{5-57}$$

Knowing the values of the first partial derivatives of the economic model and constraint equations at a feasible point, the generalized reduced gradient can be computed by Equation 5-57. This will satisfy the economic model and the constraint equations. The generalized reduced gradient is used to locate better values of the economic model in the same way unconstrained gradient search was used, i.e.

$$x_{nb} = x_{k,nb} + \alpha \nabla Y(x_k) \tag{5-58}$$

where α is the parameter of the line along the reduced gradient. A line search on α is used to locate the optimum of $Y(\mathbf{x}_{nb})$ along the generalized reduced gradient line from $\mathbf{x}_k$.

In taking trial steps as α is varied along the generalized reduced gradient line, the matrices $\mathbf{B}_b$ and $\mathbf{B}_{nb}$ must be evaluated along with the gradients $\nabla y_b(\mathbf{x}_b)$ and $\nabla y_{nb}(\mathbf{x}_k)$. This requires knowing both $\mathbf{x}_{nb}$ and $\mathbf{x}_b$ at each step. The values of $\mathbf{x}_{nb}$ are obtained from Equation 5-58. However, Equation 5-48 must be solved for $\mathbf{x}_b$; and frequently, this must be done numerically using the Newton-Raphson method. As pointed out by Reklaitis et al. (15) most of the computational effort can be involved in using the Newton-Raphson method to evaluate feasible values of the basic variables, $\mathbf{x}_b$, once the nonbasic variables have been computed from Equation 5-58. The Newton-Raphson algorithm in terms of the nomenclature for this procedure is given by the following equation.

$$\boldsymbol{x}_{i+1,b} = \boldsymbol{x}_{i,b} - \boldsymbol{B}_b^{-1}\boldsymbol{f}(\boldsymbol{x}_{i,b}, \boldsymbol{x}_{nb}) \tag{5-59}$$

where the values of the roots of the constraint equations, Equation 6-47, are being sought for $\mathbf{x}_b$, having computed $\mathbf{x}_{nb}$ from Equation 5-58. Thus, the derivatives computed for the generalized reduced gradient $\mathbf{B}_b$ matrix can be used in the Newton - Raphson root seeking procedure also.

The following example illustrates the generalized reduced gradient algorithm. It is a modification and extension of an example given by Reklaitis, et al. (15).

Example 5-11 (15)

Solve the following problem by the generalized reduced gradient method starting at point $\mathbf{x}_o$ (0,0). The constrained minimum is located at (1.2, 1.6) as shown in Figure 5-7.

minimize: $-2x_1 - 4x_2 + x_1^2 + x_2^2 + 5$

subject to: $-x_1 + 2x_2 \leq 2$

$x_1 + x_2 \leq 4$

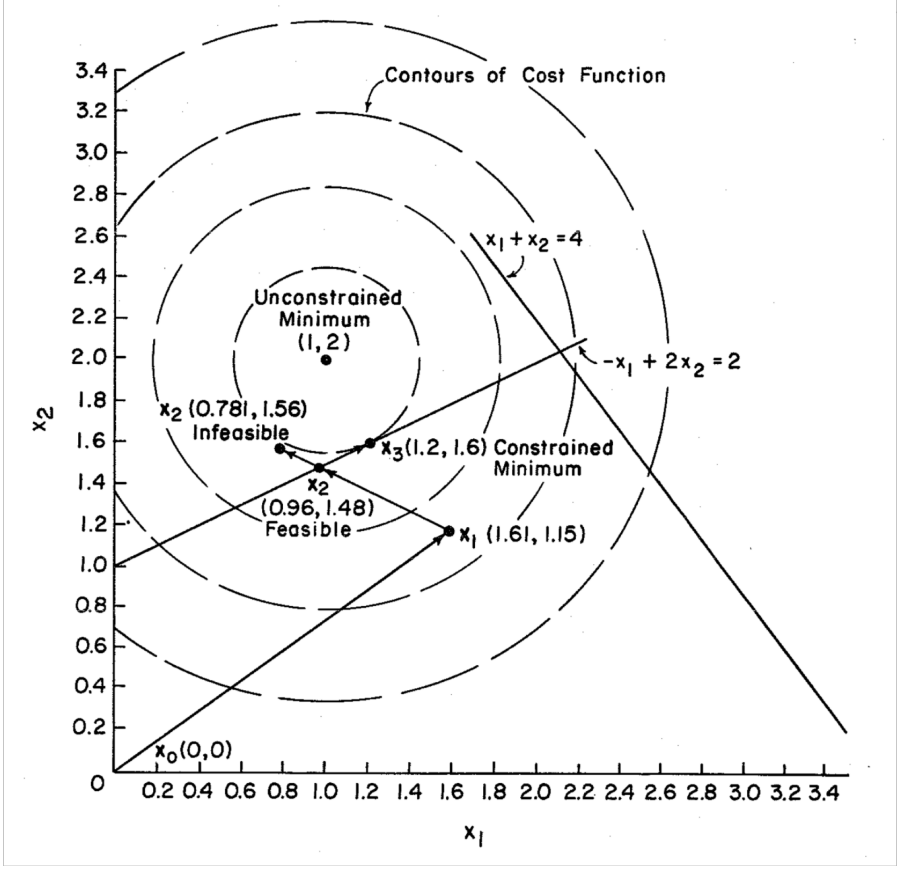


Figure 5-7 Diagram of the Solution of the Multivariable Optimization Problem in Example 5-11 by the Generalized Reduced Gradient Method

Solution: The problem is placed in the generalized reduced gradient format, Equation 5-44.

$$\text{minimize:} \quad y = -2x_1 - 4x_2 + x_1^2 + x_2^2 + 5$$

$$\text{subject to:} \quad f_1 = -x_1 + 2x_2 + x_3 - 2 = 0$$

$$f_2 = x_1 + x_2 + x_4 - 4 = 0$$

where x_3 and x_4 have been added as slack variables.

To begin x_1 and x_2 are selected to be basic variables, and x_3 and x_4 to be nonbasic variables, although others could be selected. The equation for the generalized reduced gradient is Equation 5-57 and for this problem is:

$$\begin{bmatrix} \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}_3} \\ \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}_4} \end{bmatrix}^T = \begin{bmatrix} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_3} \\ \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_4} \end{bmatrix}^T - \begin{bmatrix} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_1} & \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_2} \end{bmatrix} \begin{bmatrix} \frac{\partial \boldsymbol{f}_1}{\partial \boldsymbol{x}_1} & \frac{\partial \boldsymbol{f}_1}{\partial \boldsymbol{x}_2} \\ \frac{\partial \boldsymbol{f}_2}{\partial \boldsymbol{x}_1} & \frac{\partial \boldsymbol{f}_2}{\partial \boldsymbol{x}_2} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial \boldsymbol{f}_1}{\partial \boldsymbol{x}_3} & \frac{\partial \boldsymbol{f}_1}{\partial \boldsymbol{x}_4} \\ \frac{\partial \boldsymbol{f}_2}{\partial \boldsymbol{x}_3} & \frac{\partial \boldsymbol{f}_2}{\partial \boldsymbol{x}_4} \end{bmatrix}$$

Computing the values of the partial derivative gives:

$\frac{\partial y}{\partial x_1} = -2 + 2x_1$ $\quad \frac{\partial f_1}{\partial x_1} = -1$ $\quad \frac{\partial f_1}{\partial x_2} = 2$ $\quad \frac{\partial f_1}{\partial x_3} = 1$ $\quad \frac{\partial f_1}{\partial x_4} = 0$

$\frac{\partial y}{\partial x_2} = -4 + 2x_2$

$\frac{\partial y}{\partial x_3} = 0$ $\quad \frac{\partial f_2}{\partial x_1} = 1$ $\quad \frac{\partial f_2}{\partial x_2} = 1$ $\quad \frac{\partial f_2}{\partial x_3} = 0$ $\quad \frac{\partial f_2}{\partial x_4} = 1$

$\frac{\partial y}{\partial x_4} = 0$

The generalized reduced gradient equation becomes:

$$\begin{bmatrix} \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}_3} \\ \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}_4} \end{bmatrix}^T = \begin{bmatrix} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_3} \\ \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_4} \end{bmatrix}^T - \begin{bmatrix} -2 + 2\boldsymbol{x}_1 & -4 + 2\boldsymbol{x}_1 \end{bmatrix} \begin{bmatrix} -1 & 2 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

where

$$\boldsymbol{B}_b^{-1} = \begin{bmatrix} -1 & 2 \\ 1 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} -1/3 & 2/3 \\ 1/3 & 1/3 \end{bmatrix} \qquad \boldsymbol{B}_{nb} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The equation for the generalized reduced gradient through $\mathbf{x}_0$ (0,0) is:

$$\begin{bmatrix} \dfrac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}_3} \\ \dfrac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}_4} \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T - \begin{bmatrix} -2 & -4 \end{bmatrix} \begin{bmatrix} -1/3 & 2/3 \\ 1/3 & 1/3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2/3 \\ 8/3 \end{bmatrix}$$

The generalized reduced gradient line through starting point $\mathbf{x}_0$ (0, 0, 2, 4) is given by Equation 5-59 and for this example is:

$$x_3 = 2 + 2/3\ \alpha$$

$$x_4 = 4 + 8/3\ \alpha$$

A line search is required. The equations for x_1 and x_2 are needed in terms of x_3 and x_4 to be able to evaluate $dy/d\alpha$ since $y = y(x_1, x_2)$. Solving the constraint equations for x_1 and x_2 in terms of x_3 and x_4 gives:

$$x_2 = -1/3\ (x_3 + x_4) + 2$$

$$x_1 = 1/3\ (x_3 - 2x_4) + 2$$

Substituting to have x_1 and x_2 in terms of α gives:

$$x_2 = -1/3\ (2 + 2/3\ \alpha + 4 + 8/3\ \alpha) + 2 = -10/9\ \alpha$$

$$x_1 = 1/3\ [2 + 2/3\ \alpha - 2(4 + 8/3\ \alpha)] + 2 = -14/9\ \alpha$$

Substituting into y gives:

$$y = -2\ (-14/9)\alpha - 4(-10/9)\alpha + (-14/9\ \alpha)^2 + (-10/9\ \alpha)^2 + 5$$

$$y = 68/9\ \alpha + 296/81\ \alpha^2 + 5$$

Locating the minimum along the reduced gradient line:

$$\frac{\boldsymbol{dy}}{\boldsymbol{d\alpha}} = \frac{68}{9} + \frac{2(296)}{81}\alpha = 0$$

$$\alpha = -153/148$$

Solving for x_1, x_2, x_3 and x_4 gives:

$$x_1 = 1.608 \qquad x_3 = 1.311$$

$$x_2 = 1.149 \qquad x_4 = 1.243$$

The location of point $\mathbf{x}_1$ (1.608, 1.149, 1.311, 1.243) is shown in Figure 5-7. Also, the constraint equations are satisfied.

Now, repeating the search starting at $\mathbf{x}_1$ gives the following equation for the reduced gradient.

$$\begin{bmatrix} \dfrac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}_3} \\ \dfrac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}_4} \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T - \begin{bmatrix} -2+2(1.608) & -4+2(1.149) \end{bmatrix} \begin{bmatrix} -1/3 & 2/3 \\ 1/3 & 1/3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.973 \\ -0.243 \end{bmatrix}$$

The equations for x_1, x_2, x_3 and x_4 in terms of the parameter of the reduced gradient line are now computed as:

$$x_3 = 1.311 + 0.973\alpha$$

$$x_4 = 1.243 - 0.243\alpha$$

$$x_1 = 1.61 + 0.486\alpha$$

$$x_2 = 1.149 - 0.243\alpha$$

Using the above equations, the minimum along the reduced gradient line is located by an exact line search.

$$y = -2(1.61 + 0.486\alpha) - 4(1.149 - 0.243\alpha) + (1.61 + 0.486\alpha)^2 + (1.149 - 0.243\alpha)^2 + 5$$

Setting $dy/d\alpha$ equal to zero and solving for α gives:

$$\alpha = -1.705$$

With this value of α, the values for x_1, x_2, x_3 and x_4 are:

$$x_1 = 0.781$$

$$x_2 = 1.563$$

$$x_3 = -0.348$$

$$x_4 = 1.657$$

The point $\mathbf{x}_2$ (0.781, 1.563, −0.348, 1.657) is an infeasible point as shown in Figure 5-7. The first constraint is violated ($x_3 = -0.348$). This constraint is active, an equality; and the value of α has to be reduced to have the slack variable x_3 be equal to zero, i.e.

$$0 = 1.311 + 0.973\alpha$$

$$\alpha = -1.347$$

Recalculating x_1, x_2 and x_4 for $\alpha = -1.347$ gives:

$$x_1 = 0.955$$

$$x_2 = 1.476$$

$$x_4 = 1.57$$

The point to continue the next reduced gradient search is $\mathbf{x}_2 = (0.955, 1.476, 0, 0.157)$.

Now $\partial f_1/\partial x_3 = 0$ in the reduced gradient equation since the first constraint is an equality ($x_3 = 0$). The reduced gradient equation at $\mathbf{x}_2$ becomes:

$$\begin{bmatrix} \dfrac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}_3} \\ \dfrac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}_4} \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T - \begin{bmatrix} -2+2(0.955) & -4+2(1.476) \end{bmatrix} \begin{bmatrix} -1/3 & 2/3 \\ 1/3 & 1/3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.409 \end{bmatrix}$$

Reduced gradient line is determined as was done previously:

$$x_4 = 1.57 + \alpha(0.409)$$

$$x_3 = 0$$

$$x_1 = 0.953 - 0.273\alpha$$

$$x_2 = 1.477 - 0.136\alpha$$

In this case the reduced gradient line search will be along the first constraint, since it is now an equality constraint ($x_3 = 0$).

Solving for the optimal value of α gives:

$$y = -2(0.953 - 0.273\alpha) - 4(1.477 - 0.136\alpha) + (0.953 - 0.273\alpha)^2 + (1.477 - 0.136\alpha)^2 + 5$$

Setting $dy/d\alpha = 0$ gives $\alpha = -0.890$. Then solving for x_1, x_2 and x_4 gives:

$$x_4 = 1.57 - 0.890(0.409) = 1.20$$

$$x_1 = 0.953 - 0.273(-0.890) = 1.20$$

$$x_2 = 1.477 - 0.136(-0.890) = 1.60$$

The point $\mathbf{x}_3$ (1.20, 1.60, 0, 1.20) from the reduced gradient search is the minimum of the function as shown in Figure 5-7.

A summary of the steps is as follows:

$\mathbf{x}_0 = (0, 0, 2, 4)$

$\mathbf{x}_1 = (1.608, 1.149, 1.311, 1.243)$

$\mathbf{x}_2 = (0.781, 1.563, -0.348, 1.657)$ infeasible

$\mathbf{x}_2 = (0.955, 1.476, 0, 1.57)$ reducing α to have $x_3 = 0$

$\mathbf{x}_3 = (1.2, 1.6, 0, 1.2)$

The point $\mathbf{x}_3$ (1.20, 1.60, 0, 1.20) is the minimum of the function as shown in Figure 5-7.

The texts by Reklaitis et al. (15), Himmelblau (8) and Avriel (9) are recommended for information about additional theoretical and computational details for this method. These include procedures to maintain feasibility, i.e. the GRG, GRGS and GRGC versions, stopping criteria, relation to Lagrange multipliers, treatment of bounds and inequalities, approximate Newton - Raphson computations, and use of numerical derivatives, among others.

In the first comprehensive comparison of nonlinear programming codes was conducted by Colville (21), and the generalized reduced gradient method ranked best among 15 codes from industrial firms and universities in this country and Europe. This algorithm has been a consistently successful performer in computer programs implementing it to solve industrial problems. Lasdon (2) reported that he has a GRG code available for distribution (Professor L. S. Lasdon, School of Business Administration, University of Texas, Austin, Texas 78712), and this article lists several other sources of GRG codes.

Penalty, Barrier and Augmented Lagrange Functions: These methods convert the constrained optimization problem into an unconstrained one. The idea is to modify the economic model by adding the constraints in such a manner to have the optimum be located and the constraints be satisfied. There are several forms for the function of the constraints that can be used. These create a penalty to the economic model if the constraints are not satisfied or form a barrier to force the constraints to be satisfied, as the unconstrained search method moves from the

starting point to the optimum. This approach is related to the method of Lagrange multipliers, which is a procedure that modifies the economic model with the constraint equations to have an unconstrained problem. Also, the Lagrange function can be used with an unconstrained search technique to locate the optimum and satisfy the constraints. In addition, the augmented Lagrange function combines a penalty function with the Lagrange function to alleviate computational difficulties associated with boundaries formed by equality constraints when the Lagrange function is used alone.

These penalty function type procedures predate the previously discussed methods and have been supplanted by them. They have proved successful on relatively small problems, but the newer techniques of successive linear and quadratic programming and generalized reduced gradient were required for larger, industrial-scale problems. However, the newer techniques have incorporated these procedures on occasions to ensure a positive definite Hessian matrix and to combine the equality constraints with the profit function, which then leaves only the inequalities as constraints. The following paragraphs will review and illustrate these methods since they are used in optimization codes and as additions to the newer methods. More details are given in the texts by Avriel (9), Reklaitis, et al. (15), and Gill, et al. (6) and in the review by Sargent (33).

The penalty function concept combines two ideas. The first one is the conversion of the constrained optimization problem into an unconstrained problem, and the second is to have this unconstrained problem's solution be one that forces the constraints to be satisfied. The constraints are added to the economic model in a way to penalize movement that does not approach the optimum of the economic model and also satisfy the constraint equations. The optimization problem can be written with equality and inequality constraints as:

$$\begin{aligned} &\text{minimize:} && y(\mathbf{x}) && && (5\text{-}60)\\ &\text{subject to:} && f_i(\mathbf{x}) = 0 && \text{for } i = 1, 2, ..., h\\ & && f_i(\mathbf{x}) \geq 0 && \text{for } i = h + 1, ..., m \end{aligned}$$

By combining the economic model and constraint equations, we can form a penalty function as follows:

$$P(\mathbf{x}, r) = y(\mathbf{x}) + F[r, \mathbf{f}(\mathbf{x})] \qquad (5\text{-}61)$$

The term $F[r, \mathbf{f}(\mathbf{x})]$ is a function notation that includes the constraint equations and a penalty function parameter r as variables.

Various forms of this function F have been suggested and used with various degrees of success. Some of these forms are given in Table 5-1. Referring to the table we see that these functions are of two types, interior and exterior penalty functions. The interior penalty function requires a feasible starting point, and each step toward the optimum is a feasible point. An example of an interior penalty function with an economic model subject to inequality constraints is:

$$\text{minimize: } \boldsymbol{P}(\boldsymbol{x},\boldsymbol{r}) = \boldsymbol{y}(\boldsymbol{x}) + \boldsymbol{r}\sum_{i=h+1}^{m}\frac{1}{\left[\boldsymbol{f}_i(\boldsymbol{x})\right]^2} \tag{5-62}$$

Table 5-1. Some Forms for the Function F used to Construct the Penalty Function (9,15,26,35)

Interior penalty function forms for inequality constraints (require feasible points and also are called barrier functions), ($f_i(\mathbf{x})>0$):

$r/f_i(\mathbf{x})$	$r/[f_i(\mathbf{x})]^2$
$r\ \ln[f_i(\mathbf{x})]$	$r\left\lvert f_i(\mathbf{x})\right\rvert$ if $f_i(\mathbf{x}) < 0$, otherwise 0

Exterior penalty function forms for equality constraints $f_i(\mathbf{x})=0$

$\lvert f_i(\mathbf{x})\rvert/\mathrm{r}$	$[f_i(\mathbf{x})]^2\ /\mathrm{r}$
$[f_i(\mathbf{x})]^{2\mathrm{M}}/\mathrm{r}$ (M appositive integer)	$[f_i(\mathbf{x})]^2\ /\mathrm{r}^{1/2}$

Exterior penalty function forms for inequality constraints (feasible points are not required):

$r[f_1(\mathbf{x})]^2$ if $f_i(\mathbf{x}) < 0$, otherwise 0

Constraint x_j on: $l_j \le x_j \le u_j$

$$\boldsymbol{r}\left[\frac{2\boldsymbol{x}_j - (\boldsymbol{l}_j + \boldsymbol{u}_j)}{\boldsymbol{u}_j - \boldsymbol{l}_j}\right]^{2\boldsymbol{M}} \qquad \text{(M a positive integer)}$$

An augmented Lagrange function:

$$\boldsymbol{M}(\boldsymbol{x},\lambda,\boldsymbol{r}) = \boldsymbol{y}(\boldsymbol{x}) + \sum_{i=1}^{h}\lambda_i\boldsymbol{f}_i(\boldsymbol{x}) + \boldsymbol{r}\sum_{i=h+1}^{m}\left[\boldsymbol{f}_i(\boldsymbol{x})\right]^2$$

Interior penalty functions are applicable only to inequality constraints, and the term in Equation 5-62 with the constraints will increase as feasible points approach the boundary with the infeasible region. Consequently, the function $P(\mathbf{x}, r)$ will appear to encounter a barrier at the boundary of the feasible region. Therefore, interior penalty functions are called *barrier functions*, also. The other forms of the interior penalty function shown in Table 5-1 can be used equally as well as the one used for illustration in Equation 5-62.

The parameter r in Equation 5-62 and Table 5-1 is used to ensure convergence to the optimum and have the constraint equation be satisfied. Initially, it has a relatively large value when the search is first initiated. Then, the search is repeated with successively smaller values of r to ensure that the penalty term goes to zero, and at the optimum $P(\mathbf{x}, r \rightarrow 0) = y(\mathbf{x})$. This procedure will be illustrated subsequently. The value of r can be selected by trial and error, and normally a satisfactory starting value will be between 0.5 and 50 according to Walsh (26). Also, Walsh (26) reported a formula to compute the value of r, which involves evaluating the Jacobian matrix of the economic model and the Jacobian and Hessian matrices of the $\boldsymbol{F}$ function at the starting point.

Exterior penalty function forms start at a feasible point; and they can continue toward the optimum, even though infeasible points are generated. An example of an exterior penalty function is:

$$\text{minimize: } \boldsymbol{P}(\boldsymbol{x},\boldsymbol{r}) = \boldsymbol{y}(\boldsymbol{x}) + \boldsymbol{r}^{-1/2}\sum_{i=1}^{h}\left(\boldsymbol{f}_i(\boldsymbol{x})\right)^2 + \boldsymbol{r}\sum_{i=h=1}^{m}\left(\boldsymbol{f}_i(\boldsymbol{x})\right)^2 \tag{5-63}$$

In this form infeasible points may be generated as the unconstrained search method moves. Convergence is obtained using the parameter r, and a feasible and optimal solution will be obtained.

Exterior penalty functions used for equality constraints can be combined with interior penalty functions for inequality constraints to have what is referred to as *mixed interior-exterior penalty functions*. The one used successfully by Bracken and McCormick (36) has the form:

$$\text{minimize: } \boldsymbol{P}(\boldsymbol{x},\boldsymbol{r}) = \boldsymbol{y}(\boldsymbol{x}) + \boldsymbol{r}^{-1/2}\sum_{i=1}^{h}\left(\boldsymbol{f}_i(\boldsymbol{x})\right)^2 + \boldsymbol{r}\sum_{i=h=1}^{m}\left(\boldsymbol{f}_i(\boldsymbol{x})\right)^{-1} \tag{5-64}$$

The following example illustrates that the penalty parameter r must go to zero to arrive at the optimal solution. After this example, the results of Bracken and McCormick (36) will be summarized to illustrate the procedure of using an unconstrained search technique with a penalty function to locate the optimum of the constrained problem.

Example 5-12 (37)

Form the exterior penalty function for the following problem using the penalty parameter *r*, and use the classical theory of maxima and minima to locate the minimum. The result will include the parameter *r*. Show that it is necessary to have *r* go to zero for the optimal solution of the unconstrained problem (penalty function) to be equal to the optimal solution of the original constrained problem.

$$\text{minimize: } 2x_1^2 + 3x_2^2$$

$$\text{subject to: } x_1 + 2x_2 = 5$$

The penalty function is:

$$P(x_1, x_2, r) = 2x_1^2 + 3x_2^2 + (1/r)\,[x_1 + 2x_2 - 5]^2$$

Setting the first partial derivative with respect to x_1 and x_2 equal to 0 gives:

$$\frac{\partial P}{\partial x_1} = 4x_1 + \frac{2}{r}\,[x_1 + 2x_2 - 5] = 0$$

$$\frac{\partial P}{\partial x_2} = 6x_2 + \frac{4}{r}\,[x_1 + 2x_2 - 5] = 0$$

Solving for x_1 and x_2 gives:

$$x_1 = \frac{15}{11 + 6r} \qquad x_2 = \frac{20}{11 + 6r}$$

To have the optimal solution of the penalty function be equal to the optimal solution of constrained problem *r* must be zero, i.e.,

$$x_1 = 15/11 \qquad x_2 = 20/11$$

A solution using Lagrange multipliers will give these results, also.

When a search technique is used, a value of r must be selected which is sufficiently large to allow movement toward the optimum. As the optimum is approached successively smaller values of r must be used to have the optimum of the penalty function approach the optimum of the constrained problem. Bracken and McCormick (36) have illustrated this procedure by solving the problem shown in Figure 5-8. For this problem, a mixed penalty function was selected in the form of Equation 5-64.

Figure 5-8. The Use of a Penalty Function to Converge to the Optimum of a Constrained Problem by Bracken and McCormick (36).

Constrained problem:

$$\text{minimize:} \quad (x_1 - 2)^2 + (x_2 - 1)^2 = y$$

$$\text{subject to:} \quad -x_1^2/4 - x_2^2 + 1 \geq 0$$

$$x_1 - 2x_2 + 1 = 0$$

Unconstrained mixed penalty function problem:

$$\text{minimize: } (x_1 - 2)^2 + (x_2 - 1)^2 + r[-x_1^2/4 - x_2^2 + 1]^{-1} + r^{-1/2} [x_1 - 2x_2 + 1]^2$$

Optimal solution using SUMT program:

r	x_1	x_2	y
1.0	0.7489	0.5485	1.7691
4.0×10^{-2}	0.8177	0.8323	1.4258
1.6×10^{-3}	0.8224	0.8954	1.3976
6.4×10^{-5}	0.8228	0.9082	1.3942
2.56×10^{-6}	0.8229	0.9113	1.3935
1.024×10^{-7}	0.8229	0.9113	1.3935
4.096×10^{-9}	0.8229	0.9113	1.3935

Starting point was $\mathbf{x}_0$ (2,2) with $r = 1.0$

Analytical solution $\mathbf{x}^*[(-1 + \sqrt{7})/2 = 0.8229, (1 + \sqrt{7})/4 = 0.9114]$ and $y(\mathbf{x}^*) = 1.3935$

For the unconstrained problem to represent the constrained problem and have the same solution at the optimum, i.e. $P(\mathbf{x}^*, r) = y(\mathbf{x}^*)$, the following conditions must be satisfied:

$$\lim_{r \to 0} \left\{ r \sum_{i=1}^{h} [f_i(x)]^{-1} \right\} = 0$$

$$f_i(x) = 0 \quad \text{for } i = 1, 2, \ldots, h$$

$$\lim_{r \to 0} \left\{ r^{-1/2} \sum_{i=h+1}^{m} [f_i(x)]^2 \right\} = 0 \tag{5-65}$$

$$f_i(x) \geq 0 \qquad \text{for } i = h+1, 2, \ldots, m$$

The computational effort required to meet the requirements of Equation 5-65 is illustrated by the problem given in Figure 5-8. The search technique SUMT began at starting point $\mathbf{x}_0$ (2, 2) and arrived at the apparent optimum (0.7489, 0.5485) with a value of $r = 1.0$. The search technique was started again at point (0.7489, 0.5485) using a value for r of 4.0×10^{-2} to arrive at the apparent optimum (0.8177, 0.8323) as shown in the table in Figure 5-8. This procedure was repeated continually reducing the value for r until an acceptable result was obtained for x_1 and x_2. In this case, the values from one optimal solution to the next agreed to within four significant figures. At this point, the value of r had decreased to 4.096×10^{-9}, practically zero for the problem.

In summary, significant computational effort is required to ensure that the solution of the penalty function problem approaches the solution to the constrained problem. For the illustration, the optimization problem was solved seven times as r went from 1.0 to 4.096×10^{-9} to have a converged solution of the unconstrained problem to the constrained one. This is typical of what is to be expected when penalty functions are used.

The conventional penalty function method obtains the optimal solution only at the limit of a series of solutions of unconstrained problems (33). Consequently, exact penalty functions have been proposed that would give the optimal solution in one application of the unconstrained algorithm. Several exact penalty functions have been constructed (33); but their use has been limited since they contain absolute values that are not differentiable.

A procedure corresponding to the penalty function method has used the Lagrange function. The Lagrange function is formed as indicated in Equation 5-66 where the slack and surplus variables have been used for the inequality constraints.

$$\boldsymbol{L}(\boldsymbol{x},\lambda) = \boldsymbol{y}(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \boldsymbol{f}_i(\boldsymbol{x}) \tag{5-66}$$

In this situation an initial estimate is made for the Lagrange multipliers, and the unconstrained problem given by Equation 5-66 is solved for an apparent optimum, **x**. However, this value of **x** usually does not satisfy the constraints; and the estimated values of the Lagrange multipliers are adjusted to give a new unconstrained problem that is solved again for the apparent optimum. This procedure is repeated until the optimum is located, and the constraints are satisfied. Methods have been developed to estimate the values of the Lagrange multipliers (33) for this procedure. The following simple example illustrates this idea of having to resolve the unconstrained optimization problem with various values of the Lagrange multipliers until the constraints are satisfied.

Example 5-13

Form the Lagrange function for the following constrained problem and solve it by analytical methods for values of the Lagrange multiplier of −1/2, −1.0 and −2.0. Compare these results with the analytical solution of $x_1 = x_2 = \sqrt{(2)}/2$ and $\lambda = -\sqrt{(2)}/2$.

maximize: $y = x_1 + x_2$

subject to: $f = x_1^2 + x_2^2 - 1 = 0$

The Lagrange function is:

$$L\,(x_1, x_2, \lambda) = x_1 + x_2 + \lambda\,(x_1^2 + x_2^2 - 1)$$

Using λ = -1 the Lagrange function becomes:

$$L\,(x_1, x_2) = x_1 + x_2 + (-1)\,(x_1^2 + x_2^2 - 1)$$

Solving by analytical methods gives $x_1 = 1/2$, $x_2 = 1/2$; and using these values in the constraint gives:

$$f = (1/2)^2 + (1/2)^2 - 1 = -1/2 \neq 0$$

The other values are determined in a similar fashion, and the following table summarizes the results.

λ	x_1	x_2	f
−1/2	1	1	1
−√(2)/2	−√(2)/2	√(2)/2	0
−1	1/2	1/2	−1/2
−2	1/4	1/4	−7/8

The value of the Lagrange multiplier goes from −1 to −1/2 as the value of f goes from −1/2 to 1 with the value of $f = 0$ (constraint satisfied) at $\lambda = -\sqrt{(2)}/2$.

Using the Lagrange function is similar to using the penalty function in converting a constrained problem into an unconstrained one in the sense that the problem has to be resolved until the unconstrained problem has converged to the solution of the constrained one. There appears to be a disadvantage in using the Lagrange function because a set of Lagrange multipliers (one for each constraint) has to be adjusted while only one penalty parameter is required. However, it turns out that there are difficulties in implementing penalty functions including discontinuities on the boundaries of the feasible region (11), the Hessian matrix of the penalty function can become ill conditioned (9) and the distortion of contours as r grows smaller (15). Also, it has been found that using the Lagrange function alone has been relatively unsuccessful especially for large problems (8), except when the constraints are linear (38).

Combining penalty functions and Lagrange multipliers has proved more successful, and this technique is called the augmented Lagrange method or the method of multipliers (6, 9, 15), and the relation between the penalty parameter and the Lagrange multipliers has been reported (9, 28). The augmented Lagrange function can be written as follows (11).

$$\boldsymbol{M}(\boldsymbol{x},\lambda,\boldsymbol{r}) = \boldsymbol{y}(\boldsymbol{x}) + \sum_{i=1}^{h} \lambda_i \boldsymbol{f}_i(\boldsymbol{x}) + \boldsymbol{r} \sum_{i=h+1}^{m} \left[\boldsymbol{f}_i(\boldsymbol{x})\right]^2 \tag{5-67}$$

and an algorithm for updating the Lagrange multipliers has been given by Avriel (11).

$$\lambda_{i,k+1} = \lambda_{i,k} - r\, f_i(\mathbf{x}_k) \tag{5-68}$$

Avriel (11) has given an example of the use of this procedure for a simple problem. There have been difficulties associated with this method in the choice of the penalty parameter r. As discussed by Gill, et al. (6), too small a value can lead to an unbounded number of unconstrained searches having to be performed, in addition to a possible ill-conditioned Hessian matrix of the Lagrange function. As will be seen in the section on comparison of techniques, these methods have not performed as well as successive linear and successive quadratic programming and the generalized reduced gradient method.

Other Multivariable Constrained Search Methods: Other methods for constrained multivariable problems fall into a class referred to as *feasible directions, projection methods* or *methods of restricted movement*. Also, there are random search procedures, cutting plane methods and feasible region elimination techniques. The concepts associated with each of these procedures are described and references given for sources of more information. These techniques have founded limited application, and the reasons for this are described.

Restricted Movements: These methods are described in some detail by Avriel (9, 11) and others (6, 8, 15, 27). According to Reklaitis et al. (15) even though there are similarities between these projection methods and reduced gradient techniques, the latter are preferred because sparse-matrix methods can be used but the former methods are said to have "sparsity-destroying matrix products." Consequently, the details of these methods are available in the previously cited references, and only an illustration from McMillan (27) will be given to show some of the concepts involved.

A simple problem is shown in Figure 5-9 where the starting point is in the feasible region at point $\mathbf{x}_0$ (8, 2). There are three constraints that bound the feasible region, and the unconstrained maximum lies outside of the region. This gradient-projection method begins by a single variable search along the gradient line to locate a maximum. The maximum along the line will be found where a constraint is encountered at point $\mathbf{x}_1$(6.6, 3.6). The gradient line at point $\mathbf{x}_1$ points into the infeasible region. Therefore, to continue to move toward the optimum, the gradient line is projected on the constraint, and the search proceeds in this projected-gradient direction along the constraint. The constraint is linear, and a single variable search for the maximum locates point $\mathbf{x}_2$ (6,4) that is the intersection with another constraint. The gradient line at $\mathbf{x}_2$ (6, 4) points into the infeasible region so it is projected on the constraint, in this case $x_2 = 4$; and the single variable search for the maxima continues. The search arrives at point $\mathbf{x}_3$ (5.5, 4), which is the constrained maximum.

In summary, the procedure began with an unconstrained search method, gradient search, until a constraint was encountered. The unconstrained search line was projected on the constraints to be able to stay in the feasible region, and it moved until the maximum was located. Other unconstrained search methods could have been used rather than gradient search, such as the BFGS method. Also, had the constraints been curved, the search method would have difficulty following the constraint; and a hemstitching pattern would have developed as the search method attempted to follow the active nonlinear constraint. This pattern is illustrated in Figure 5-10 and it is one of the problems encountered with this method, as discussed by Avriel (9).

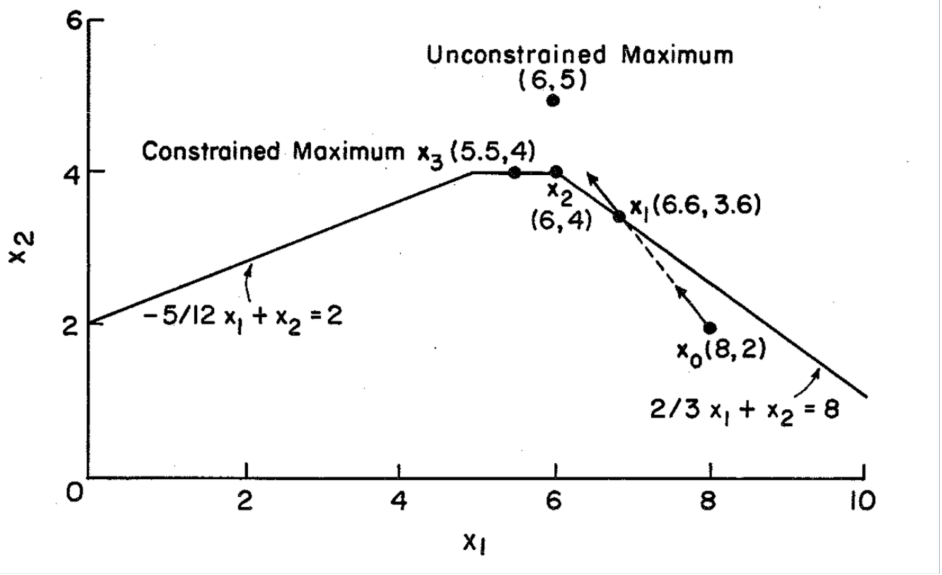


Figure 5-9 Illustration of the Gradient Projection Method, after McMillian (27)

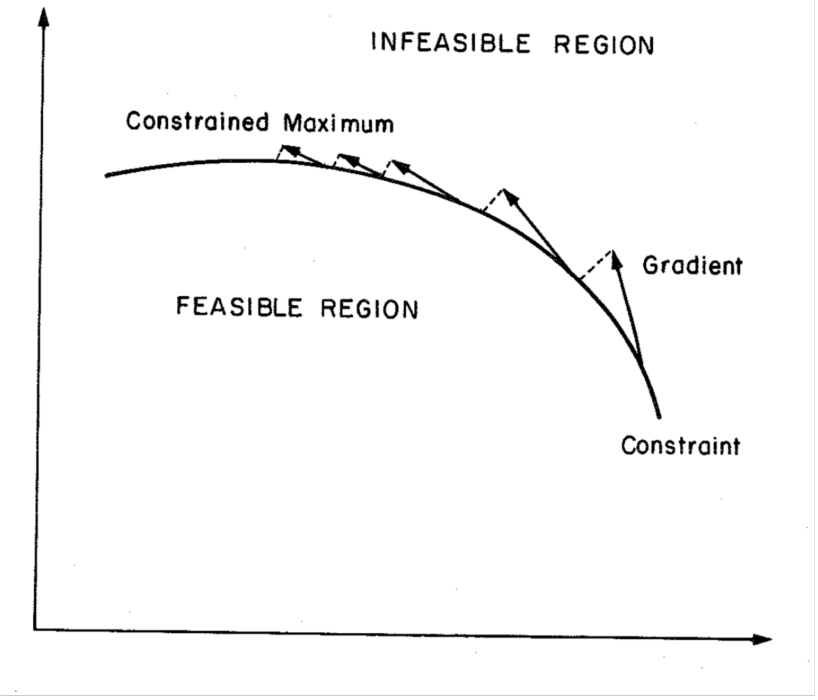


Figure 5-10 Hemstitching Pattern Developed by Restricted Movement Methods Following an Active Constraint, after Avriel (9)

Cutting Plane Methods: In these methods (15, 28), the nonlinear optimization problem is formulated as follows. Beginning with the nonlinear optimization problem as:

$$\text{minimize: } y(\mathbf{x}) \tag{5-69}$$

$$\text{subject to: } f_i(\mathbf{x}) \geq 0 \quad \text{for } i = 1, 2, ..., m$$

The problem is converted to the following one:

$$\text{minimize: } x_0$$

$$\text{subject to: } f_i(\mathbf{x}) \geq 0 \tag{5-70}$$

$$x_0 - y(\mathbf{x}) \geq 0$$

which gives a linear economic model. Then, if a starting point is selected that violates only one of the constraints, this constraint can be linearized; and the resulting problem can be solved by linear programming. At the new point, the most violated constraint is added, and linearized to find the third point in the search. The procedure continues adding constraints until the optimum is reached and the constraints are satisfied. However, a number of computational difficulties have been encountered with this procedure according to Avriel (9); but it has been attractive because convergence to the global optimum is guaranteed, if the economic model and constraints are convex functions.

Random Search: In random search, the feasible region is divided into a grid where each nodal point is considered to be the location of a point to compute the value of the economic model, i.e. an experiment. Then if an exhaustive search is performed by calculating the value of the economic model at each point in the grid, these experiments could be ranked from the one with the maximum value of the economic model to the one with the minimum value. However, a specified number of these experiments could be selected randomly, and the value of the economic model evaluated at the points. It would be possible to make a statement about the point with the largest value of the economic model being in a certain best fraction of all of the experiments with a specific probability. For example, if there were 1000 nodal points, and if one experiment was placed randomly in these points, the probability of choosing one in the top 10% would be 100/1000 = 0.1. Also, the probability of not choosing one in the top 10% would be 1 - 0.1 = 0.9. (Probability is defined as the relative frequency of occurrences of an event.)

If two experiments were placed randomly in the grid on the feasible region, the probability of not finding one in the top 10% would be $(0.9)^2 = 0.81$, and the probability of one of these two being in the top 10% is $1 - 0.81 = 0.19$. Continuing, after n trials the formula is:

$$p(0.1) = 1 - (0.9)^n \tag{5-71}$$

For $n = 16$, the probability of finding one of these 16 experiments to be in the best fraction of 0.1 would be $p\ (0.1) = 0.80$. For $n = 44$, $p\ (0.1) = 0.99$ which is almost a certainty.

The generalization of this procedure, Wilde (10), is given by the following equation.

$$p\,(f) = 1 - (1 - f)^n \tag{5-72}$$

In this equation $p\,(f)$ is the probability of finding at least one nodal point in the best fraction, f, having placed n experiments randomly in the feasible region. Several values of n have been computed by Wilde (10) having specified f and $p(f)$. These are given in Table 5-2. Equation 5-72 was used in the following form for these calculations.

$$n = ln\,[1 - p(f)]/ln\,(1-f) \tag{5-73}$$

Table 5-2. The Number of Experiments, n, Required to Have at Least One in the Best Fraction, f, with a Probability, $p(f)$, having a Total of 1000 Possible Experiments, after Wilde (10).

	$p(f)$			
f	0.80	0.90	0.95	0.99
0.1	16	22	29	44
0.01	161	230	299	459
0.005	322	460	598	919

Referring to Table 5-2, 16 experiments would be required to have at least one in the top 10% with a probability of 0.80 from a total of 1000 experiments. To have at least one value of the economic model in the top 0.5% with a probability of 0.99, 919 experiments of the total of 1000 would have to be measured, i.e. the economic model would have to be evaluated at almost all of the nodal points. Also, it should be noted that the values for n reported in the table have been rounded off, e.g. 919 is 918.72... computed from Equation 5-73. If there had been a total of 100 experiments 92 would have been required for at least one in the top 0.5 % with a probability of 0.99.

The number of nodal points is somewhat independent of the number of variables in the economic model. Also, the results are independent of the number of local maxima or minima. These two facts are considered to be the important advantages of random search. This has led to adaptive random search where a random search is conducted on part of the feasible region. Then another section of the feasible region is selected which contained the largest value of the economic model to repeat the placing of another set of random measurements. This converts random search into a search technique, and it has been called *adaptive random search* by Gaddy and co-workers (40, 41).

Using this technique Martin and Gaddy (41) have described the optimization of a maleic anhydrate process. They showed that their adaptive randomly directed search method efficiently optimized the types of problems described as large, heavily constrained and often containing mixed integer variables.

Feasible Region Elimination Techniques: These methods are described in some detail by Wilde and Beightler (12) and are an extension of the ideas associated with the interval elimination, single variable search methods. Two techniques are contour tangent elimination and multivariable dichotomous elimination. The first method is applicable only to functions that are strongly unimodal; and the second procedure requires that functions be rectangularity unimodal, which is more restrictive than strongly unimodal.

A strongly unimodal function has a strictly rising path from any point in the feasible region to the optimum. Consequently, a function with a curving ridge would not be strongly unimodal. An example of a strongly unimodal function is given in Figure 5-11. The line from point A to the maximum illustrates a strictly rising path.

The multivariable elimination technique is illustrated in Figure 5-11 for two independent variables. First, a starting point, $\mathbf{x}_0$, in the feasible region is selected; and a contour tangent line is determined. The area below the contour tangent can be eliminated since it does not contain the optimum; and the procedure continues by placing another experiment in the area that contains the optimum, e.g. point $\mathbf{x}_1$. Measuring the contour tangent at $\mathbf{x}_1$, an additional region can be eliminated. In this case it will be above the contour tangent line, and the region that contains the maximum is reduced. Again, another measurement is placed in the remaining area that contains the optimum, e.g. $\mathbf{x}_3$; and the contour tangent is determined. Eliminating the area to the left of this contour tangent, now the region that contains the optimum has been reduced to the triangular shaped area bounded by the three contour tangents as shown in Figure 5-11. The procedure continues in this fashion until the region that contains the optimum has been reduced to a satisfactory size. The details of the computational procedure are given by Wilde and Beightler (12), and the method has had limited use because of the restrictive requirement of being applicable only to strongly unimodal functions.

There are a number of other methods that could have been mentioned, all of which have had some degree of success in optimizing industrial problems, and these are described in the references at the end of this chapter. Many of these methods are modifications and/or combinations of the procedures that have been discussed. In the next section comparisons will be given of the performance of constrained multivariable procedures.

Comparison of Constrained Multivariable Search Methods: The evaluation of the effectiveness of constrained multivariable optimization procedures depends on several interrelated things. These are the optimization theory, the algorithm or the combination of algorithms to implement the theory, the computer program and programming language used for computations with the algorithms, the computer to run the program and the optimization problems being solved. In comparing constrained optimization procedures, usually the same optimization problems are

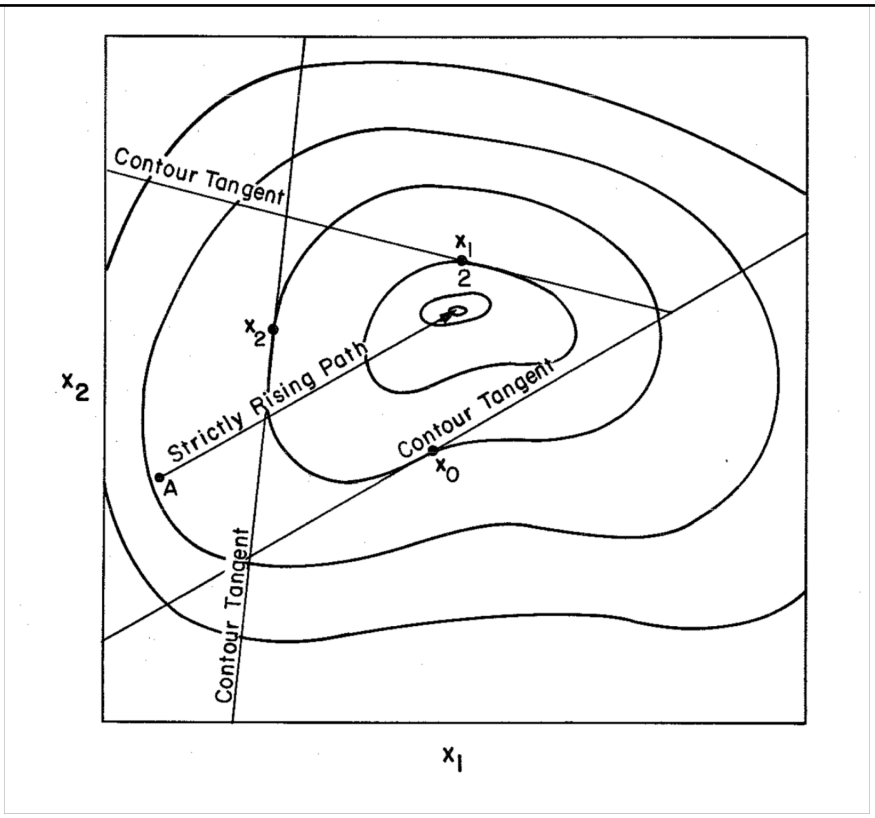


Figure 5-11 Illustration of the Method of Contour Tangents, after Wild and Beightler (12)

solved; and comparisons are based on measurements of computer time or the number of times the economic model and constraints are evaluated to come within a certain fraction of the optimum. If different computers are used to solve the optimization problems, then a timing program such as a matrix inversion is run on each machine to give a point of comparison among the computers. Consequently, if there is a superior optimization algorithm, the other factors that affect performance have made it difficult to detect.

There is debate about which algorithms and/or computer codes are the better ones, and Lasdon (3) recommended having available several computer codes that incorporate the more successful methods. A judgment about the ones to have can be obtained from the following reviews of industrial experience reporting the use of optimization procedures on process and plant problems.

In an Exxon study by Simon and Azma (1) fifteen industrial optimization problems were solved using four established optimization codes, and their results are summarized in Table 5-3. The fifteen problems had from 5 to 250 variables; and there were a number of active constraints at the optimum, ranging for each size problem from 50 to 95% of the number of variables.

Two of the four optimization codes, ECO and SLP, were developed by Exxon. The ECO program used successive quadratic programming with many of the features described previously for the Wilson, Han, Powell algorithm including a Davidon, Fletcher, Powell update for the Hessian matrix. The SLP program used successive linear programming as described previously with enhancements to speed convergence and circumvent problems with infeasibilities. The GRG2 program used the generalized reduced gradient method, and this program was developed by Lasdon (3). The MINOS program used a projected augmented Lagrange algorithm combined with the generalized reduced gradient algorithm, and this program was developed by Murtagh and Saunders (43).

The problems were run on Exxon's IBM 3033 computer, and the key results extracted from the article are given in Table 5-3. These include the average and range of the number of function calls and the average CPU time used. The optimization applications were complex simulations, and numerical differentiation was required. Consequently, the number of times that the economic model and constraints were evaluated (function calls) was viewed as the primary indicator of performance. CPU times were said to be a guide to performance and were not available for SLP optimizations. Also, two termination criteria were used in the various convergence tests of the programs to have the value of the economic model and constraints to be within the tolerance of 0.001 and the economic model to be until 0.1% of the optimum.

In reviewing the results in Table 5-3, SLP and MINOS solved all of the test problems. It was reported by Simon and Azma (1) that the performance of SLP was better on more tightly constrained problems. Also, they reported that MINOS had impressively fast run times from the use of sparse matrix computational features. The GRG2 code solved all of the 5- and 20-variable problems and three of the six 100- variable problems. This code required the greatest number of function calls compared to the others. The ECO code solved all of the 5- and 20- variable problems, the two 100-variable, linearly constrained problems and one of the 100-variable, nonlinearly constrained problems. It was reported that nonlinear constraints caused numerical difficulties for the ECO code, because it did not contain special error checking and matrix inversion features of the other codes.

This study has shown that to solve large industrial problems these three optimization algorithms must be supplemented with other features associated with numerical differentiation and sparse matrix manipulations. In the following review of other comparison studies of optimization codes, the three procedures, SLP, SQP and GRG, were found to be superior to others, and the relative merits of these methods have been tabulated by Lasdon and Warren (22).

Successive linear programming (SLP) was said to be easy to implement, widely used in practice, rapidly convergent when the optimum is at a vertex, and it was able to solve very large problems. Furthermore, it did not attempt to satisfy equalities at each iteration, may converge slowly on problems with non-vertex optimum and will violate nonlinear constraints until convergence is reached.

Table 5-3 Comparisons of ECO, GRG2, MINOS and SLP from the Exxon Evaluation Using 29 Optimization Problems from Simon and Azma (1).

Optimization Problems Number of Variables (Number of Problems)	Average Number of Function Calls (Range) CPU Seconds ECO	GRG2	MINOS	SLP
Convergence tolerance of 0.001				
5 variables (7)	32(23-47) 0.17	87(33-203) 0.14	33(15-49) 0.28	73(44-94) NR*
20 variables (12) 261)	43(29-67) 5.0	537(475-672) 3.0	166(46-263) 0.95	181(111- NR
100 variables (2) linear constraints	51(50 & 52) 42.0	445(1 problem) 332.0	48(46 & 50) 1.8	69(57-80) NR
100 variables (4) nonlinear constraints	Failed	2005(445 & 5225) (2 problems only) 983.0	145(NR) 2.8	103(NR) NR
250 variables (4) 181)	Not Run	Not Run	881(747-1073) 25.0	131(105- NR
Convergence to 0.1% of optimum				
5 variables (7)	16(12-24) 0.1	73(25-174) 0.1	23(15-29) 0.2	22(12-34) NR
20 variables (12) 229)	29(18-57) 3.7	486(311-647) 2.5	145(44-252) 0.8	131(62- NR
100 variables (2) linear constraints	25(21 & 28) 25.0	397(1 problem) 289.0	47(46 & 48) 01.8	19(14-24) NR
100 variables (4) nonlinear constraints	Failed	1682(606 & 2758) (2 problems only) 44.60	162(82-208) 2.7	47(20-76) NR
250 variables (4) 175)	Not Run	Not Run	841(714-1062) 24.0	83(47- NR

*NR - Not Reported

Successive quadratic programming (SQP) is said to require the fewest function and gradient evaluations of the three methods. It did not attempt to satisfy equalities at each iteration and will violate nonlinear constraints until convergence is reached. It is more difficult to implement than SLP and requires a good quadratic programming solver.

Generalized reduced gradient (GRG) is said to be probably the most robust and versatile, being able to employ existing process simulators using the Newton-Raphson method. It is the most difficult to implement and needs to satisfy equality constraints at each step of the algorithm.

In a dissertation by Sandgren (20) on the utility of nonlinear programming algorithms, 35 optimization algorithms were collected from university and industry sources of which 29 used penalty functions, four used generalized reduced gradient (GRG) and two used successive linear programming (SLP). Thirty test problems were selected from a variety of applications and sources that had from 2 to 48 variables and 4 to 75 constraints. Computations were performed on Purdue University's CDC 6500 computer in double precision, and all gradients were calculated using a forward difference approximation. Solution times were measured, and a rating procedure was used to rank the programs. The results showed that the four codes using the GRG algorithm and one code using SLP solved 50% of the test problems using 25% or less of the computer time averaged for all of the programs. This study established fairly conclusively that GRG and SLP algorithms are superior to penalty function methods.

In a study by Schittkowski reported by Reklaitis, et al. (15) 22 optimization programs and 180 test problems were evaluated. The optimization program included 11 SLP, 3 GRG, 4 SQP and 4 penalty function codes; and nine criteria were used and weighted to rank the programs. The ranking of the algorithm classes were in the order of SQP, GRG, and SLP with penalty functions last. Also, it was emphasized that these tests showed that code reliability is more a function of the programming of the algorithm than the algorithm itself.

In probably the first comprehensive study of nonlinear constrained optimization procedures, Colville (21) organized participants from fifteen industrial firms and universities and had them test eight industrial problems with their 30 optimization codes. He grouped the methods into five categories and developed a scoring procedure. This involved using a timing program for matrix inversion since the results were obtained from a number of different computers. The highest score was received by the GRG method.

Himmelblau (8) extended these results and ran some of Colville's and other problems on the same computer. Again, the GRG code was the best performer. However, Palacios-Gomez et al. (47) have shown that their improved version of SLP based on industrial computational experience was comparable to or better than GRG2 and MINOS on Himmelblau's and other test problems.

Successive quadratic programming and generalized reduced gradient algorithms have been used with large computer simulations and flowsheeting programs for optimal design. Biegler and Hughes (44, 49) showed that successive quadratic programming was effective for optimization of a propylene chlorination process simulation. In the previously mentioned study of Jirapongpham, et al. (42) it demonstrated that the WHP algorithm was effective for process flowsheeting optimization. Locke and Westerberg (45, 46) used an advanced quadratic programming algorithm

with an equation-oriented process flow-sheeting program with success. Chen and Stadtherr (48) reported enhancements of the WHP method that were effective on several chemical process optimization problems. Biegler and Cuthrell (53) showed the Armijo line search to be one of several improvements to successive quadratic programming. Drud (58) has developed a GRG program CONOPT that used the industry standard MPS input format for large static and dynamic problems at the World Bank.

In summary, the three methods of choice for optimization of industrial scale problems are successive linear and successive quadratic programming and the generalized reduced gradient method. The available programs that use these procedures are elaborate and use a combination of techniques for efficient computer computations. Sources for programs using these methods are given by Waren and Lasdon (2), Reklaitis et al. (15) and Gill, et al. (6). Waren and Lasdon (2) list the desirable features of nonlinear programming software that can be used as a guide for selection of codes.

The GAMS (General Algebraic Modeling System) programming language was developed by the GAMS Development Corporation 1217 Potomac Street, NW, Washington, D.C. 20007 (http://www.gams.com). GAMS is a high-level modeling language for mathematical programming and optimization. It consists of a language compiler and a stable of integrated high-performance optimization programs called "solvers." GAMS model types include Linear Programming (LP), Mixed-Integer Programming (MIP), Mixed-Integer Non-Linear Programming (MINLP), and different forms of Non-Linear Programming (NLP). There are over 30 solvers (optimization codes) that can be selected to solve these programming problems. GAMS is available for use on personal computers, workstations, mainframes and supercomputers. Note, "programming," means "scheduling" and not "computer programming."

GAMS Distribution 26.1.0 (February 2, 2019) is currently available for download from the GAMS web site www.GAMS.com without charge. GAMS will operate as a free demo system without a valid GAMS license. The model limits in demo mode are 300 constraints and variables, 2000 nonzero elements, (of which 1000 can be nonlinear), 50 discrete variables (including semi continuous, semi integer and member of SOS-Sets) with additional global solver limits of 10 constraints and variables. There are the installation notes for Windows, Mac, and UNIX. The GAMS distribution includes the GAMS Manuals in electronic form, and hard copies can be ordered through Amazon.

Stochastic Approximation Procedures

All of the procedures for deterministic processes can be confounded by random error. There are search techniques that converge to an optimum in the face of random error, and some of these will be discussed briefly following the approach of Wilde (10) who gives more details about these methods. Random (e.g. experimental) error clouds the perception of what is happening and greatly hampers the search for the optimum. Stochastic approximation procedures deal with random error as noise superimposed on a deterministic process. Therefore, convergence to the optimum must be considered first, and then efficiency can be evaluated. The works of Dvoretzky, Kiefer and Wolfowitz in this area have been summarized in an excellent manner by Wilde (10).

Consequently, only the most important of these techniques will be described. This is the Kiefer-Wolfowitz stochastic approximation procedure, and it is applicable for n independent variables.

With noise present a search technique is forced to creep to prevent being confounded by random error. However, for unimodal functions, it can be shown that stochastic approximation procedures converge to the optimum in the mean square and with probability one (10).

The Kiefer-Wolfowitz algorithm is given by the following equation (similar to steep ascent). Beginning at a starting point $\mathbf{x}_0$, the method proceeds according to this equation:

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \\ \vdots \\ x_{n,k+1} \end{bmatrix} = \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ \vdots \\ x_{n,k} \end{bmatrix} + \frac{a_k}{c_k} \begin{bmatrix} y(x_{1,k}+c_k, x_{2k}, \ldots, x_{n,k}) - y(x_{1,k}-c_k, x_{2k}, \ldots, x_{n,k}) \\ y(x_{1,k}, x_{2k}+c_k, \ldots, x_{n,k}) - y(x_{1,k}, x_{2k}-c_k, \ldots, x_{n,k}) \\ \vdots \\ y(x_{1,k}, x_{2k}, \ldots, x_{n,k}+c_k) - y(x_{1,k}, x_{2k}, \ldots, x_{n,k}-c_k) \end{bmatrix} \tag{5-74}$$

For convergence, the parameters a_k and c_k must satisfy the following criteria

$$\begin{aligned} &\lim_{k \to \infty} a_k = 0 \\ &\lim_{k \to \infty} c_k = 0 \\ &\sum_{k=1}^{\infty} a_k = \infty \\ &\sum_{k=1}^{\infty} (a_k / c_k)^2 < \infty \end{aligned} \tag{5-75}$$

The following example illustrates the use of the Kiefer-Wolfowitz procedure.

Example 5-13

Develop the procedure to obtain the minimum of a function of the form that is affected by experimental error.

$$Ax_1(x_1 - x_1^*)^2 + Bx_2(x_2 - x_2^*)^2$$

The value of the minimum is somewhere on the interval:

$$1 \le x_1^* \le 3 \qquad 1 \le x_2^* \le 3$$

Starting with the mid-point of the interval, give the equations for the second, third and last of twenty trials.

Solution: $a_k = 1/k$, $c_k = 1/k^{1/4}$ satisfies the criterion of the Equation 5-75.

For $\mathbf{x}_2 = (x_{1,2}, x_{2,2})$, $k = 1$:

$$\begin{bmatrix} x_{1,2} \\ x_{2,2} \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \begin{bmatrix} y(3,2) - y(1,2) \\ y(2,3) - y(2,1) \end{bmatrix}$$

For $\mathbf{x}_3 = (x_{1,3}, x_{2,3})$, $k = 2$:

$$\begin{bmatrix} x_{1,3} \\ x_{2,3} \end{bmatrix} = \begin{bmatrix} x_{1,2} \\ x_{2,2} \end{bmatrix} + \frac{1}{2^{3/4}} \begin{bmatrix} y(x_{1,2} + 2^{-1/4}, x_{2,2}) - y(x_{1,2} - 2^{-1/4}, x_{2,2}) \\ y(x_{1,2}, x_{2,2} + 2^{-1/4}) - y(x_{1,2}, x_{2,2} - 2^{-1/4}) \end{bmatrix}$$

For $\mathbf{x}_{20} = (x_{1,20}, x_{2,20})$, $k = 19$:

$$\begin{bmatrix} x_{1,20} \\ x_{2,20} \end{bmatrix} = \begin{bmatrix} x_{1,19} \\ x_{2,19} \end{bmatrix} + \frac{1}{19^{3/4}} \begin{bmatrix} y(x_{1,19} + 19^{-1/4}, x_{2,19}) - y(x_{1,19} - 19^{-1/4}, x_{2,19}) \\ y(x_{1,19}, x_{2,19} + 19^{-1/4}) - y(x_{1,19}, x_{2,19} - 19^{-1/4}) \end{bmatrix}$$

There are variations of the above procedure such as using only the sign of the approximation to the derivatives. This can be used effectively when there is difficulty with convergence that is being caused by the shape of the curve on either side of the optimum. Also, a forward difference approximation can be used in evaluating the derivative rather than the central difference form, but convergence is not as rapid.

Closure

In this chapter the important algorithms for optimizing a nonlinear economic model with nonlinear constraints have been described, and their performance has been reviewed. This required presenting methods for unconstrained problems first and outlining the strategy required to move from a starting point to a point near an optimum. It was not possible to discuss each of the many algorithms that have been proposed and employed as unconstrained multivariable search techniques, but the references at the end of the chapter will lead to comprehensive descriptions of those procedures. The texts by Avriel (9), Fletcher (4, 5), Gill et al. (6), Himmelblau (8), McCormick (7) and Reklaitis et al. (15) are particularly recommended for this purpose. However, the more successful algorithms were described for both unconstrained and constrained optimization problems. It is recommended that the BFGS algorithm be used for unconstrained problems and a Fortran program for this procedure has been included at the end of the chapter, Table 5-4. For constrained problems the three methods that have been more successful in

comparison studies on industrial problems are successive linear and quadratic programming (SLP and SQP) and generalized reduced gradient method (GRG). Advanced computation techniques for numerical derivatives and sparse matrix manipulations are required to have efficient codes, and sources to contact for these types of programs were referenced.

In addition to deterministic optimization methods, stochastic approximation procedures were described briefly based on material from Wilde's book (10). These methods are designed to locate the optimum in the face of experimental error, even though their movement is slowed to avoid being confounded by random and gross errors.

This area of optimization is probably the most rapidly growing part of the subject. The growth of computers and applied mathematical techniques for the solution of large systems of equations promises to continue to allow significant developments to take place.

Table 5-4. FORTRAN Program with Sample Input and Output for BFGS Search of an Unconstrained Nonlinear Function

```
C       PROGRAM BFGS
C-------------------------------------------------------------------------------------------------------
C       NOTATION :
C       NTERM       : NO. OF INDEPENDENT VARIABLES IN THE COST FUNCTION
C       X           : INDEPENDENT VARIABLES
C       EPS         : STOPPING CRITERION ON COST FUNCTION
C       ITER        : LOOP COUNTER
C       HESS        : HESSIAN MATRIX
C       K           : PARAMETER OF THE LINE SEARCHED
C-----------------------------------------------------------------------
C
        INTEGER ITER
        DOUBLE PRECISION TOLER, FUNCT, FIBON,
        1   HESS(20, 20), GRAD(20), GRAD1(20), GAMMA(20), DELTA(20),
        2   HG(20), K, ERR, ERROLD, EPS, GPHG, DPG, X(20), S(20)
C
        COMMON X, S, NTERM
C
        ITER   = 0
        K      = 0
C-----------------------------------------------------------------------
C       READ AND ECHO INPUT DATA
C-----------------------------------------------------------------------
C
        READ(5,*) NTERM, EPS
        READ(5,*) ( X(I), I=1,NTERM)
        WRITE(6,600) NTERM, EPS,( X(I), I=1,NTERM)
 600  FORMAT(/,5X,'INPUT DATA :  ',
```

```
     &        /,5X,'NO. OF INDEPENDENT VARIABLES, NTERM    = ',I4,
     &        /,5X,'STOPPING CRITERION, EPS                = ',F9.4,
     &        /,5X,'STARTING POINTS, X                     = ',10(1X,F6.2))
      WRITE(6,601)
601   FORMAT(/,5X,'RESULTS    :',
     &        /,5X,'ITERATION',2X,'COST FUNCTION',6X,'VALUES OF X',12X,'
     &K',/)
C-----------------------------------------------------------------------
C       BFGS SEARCH
C-----------------------------------------------------------------------
        ERR= FUNCT( X )
        CALL PRINT ( ITER, NTERM, ERR, X, K)
        CALL SLOPE( GRAD, ERR )
C-----------------------------------------------------------------------
C       FORM THE IDENTITY MATRIX
C-----------------------------------------------------------------------
        DO 40 I=1, NTERM
          DO 40 J=1, NTERM
             IF (I.NE.J) HESS(I,J)= 0.0
             IF (I.EQ.J) HESS(I,J)= 1.0
40      CONTINUE
30      CONTINUE
C
        ERROLD= ERR
        ITER= ITER + 1
C-----------------------------------------------------------------------
C       S (I) = HESSIAN*GRADIENT
C-----------------------------------------------------------------------
        DO 50 I=1, NTERM
          S (I)= 0.0
          DO 50 J=1, NTERM
             S (I)= S(I) + HESS (I, J) * GRAD (J)
50           CONTINUE
C-----------------------------------------------------------------------
C       K = ALPHA IN EQN.6-17
C-----------------------------------------------------------------------
        K= FIBON(DUMMY)
C-----------------------------------------------------------------------
C       DETERMINE NEXT X VALUE WITH EQN.6-17
C       DELTA = ALPHA*HESSIAN*GRADIENT, IN EQN. 6-17
C-----------------------------------------------------------------------
        DO 60 I=1, NTERM
          DELTA (I)= K * S (I)
          X(I)= X(I) – DELTA (I)
60        CONTINUE
```

```
      ERR= FUNCT(X)
      CALL SLOPE( GRAD1, ERR )
C-----------------------------------------------------------------------
C     DETERMINE NEW BFGS MATRIX WITH EQN.6-20
C-----------------------------------------------------------------------
      DPG=0.0
      DO 70 I = 1, NTERM
        GAMMA (I)= GRAD1 (I) – GRAD (I)
        DPG = DPG + GAMMA (I) * DELTA (I)
 70     CONTINUE
      DO 80 I=1, NTERM
        GRAD (I) = GRAD1 (I)
 80     CONTINUE
      GPHG= 0.0
      DO 90 I=1, NTERM
        HG(I)= 0.0
        DO 90 J=1,NTERM
          HG (I) = HG (I) + HESS (I, J) * GAMMA (J)
          GPHG = GPHG+ HESS (I, J) * GAMMA (I) * GAMMA (J)
 90     CONTINUE
      DO 100 I = 1, NTERM
        DO 100 J = 1, NTERM
          HESS (I, J) = HESS (I, J) - (HG (I) * DELTA (J) / DPG)
     $       - (DELTA (I) * HG (J) / DPG)
     $       + (1 + (GPHG / DPG)) * (DELTA (I)
     $       * DELTA (J) / DPG)
 100    CONTINUE
      TOLER = DABS (ERR - ERROLD)
      IF (TOLER .GE. EPS) CALL PRINT( ITER, NTERM, ERR, X, K)
      IF (TOLER .GE. EPS) GO TO 30
      STOP
      END
C-----------------------------------------------------------------------
C     COMPUTATION OF PARTIAL DERIVATIVES
C-----------------------------------------------------------------------
      SUBROUTINE SLOPE( DERIV, E )
      DOUBLE PRECISION DERIV (20), E, DELTA, TEMPX, Y, X(20),S(20), FUNCT
      COMMON X, S, NTERM
C
      DO 30 I=1, NTERM
        DELTA= 1.0E-04
        TEMPX= X(I)
        X(I)= X(I) + DELTA
        Y= FUNCT( X )
        DERIV(I)= (Y - E)/DELTA
```

```
          X(I)= TEMPX
 30       CONTINUE
        RETURN
        END
C-----------------------------------------------------------------------
C       PRINT RESULTS
C-----------------------------------------------------------------------
        SUBROUTINE PRINT( I, N, VAL, X, K)
        DOUBLE PRECISION X(20), VAL, K
        WRITE(6,600) I,VAL,(X(J),J=1,N), K
 600    FORMAT(7X,I3,6X,F10.3,4X,10(1X,F7.3))
        RETURN
        END
C-----------------------------------------------------------------------
C       FIBONNACCI SEARCH FUNCTION
C-----------------------------------------------------------------------
C       LBOUND      : LOWER BOUND
C       HBOUND      : UPPER BOUND
C       INTER       : INITIAL INTERVAL
C       FINTER      : FINAL INTERVAL
C       RATIO       : RATIO OF INITIAL AND FINAL INTERVALS
C       DELTA       : DISPLACEMENT OF AN EXPERIMENT FROM THE BOUNDARY,
C                     EQN.5-44, INITIALLY
C       FIBO        : FIBONNACCI NUMBERS
C       FACT        : FIBO(N+1)/FIBO(N-1)
C-----------------------------------------------------------------------
        DOUBLE PRECISION FUNCTION FIBON( DUMMY )
C
        DOUBLE PRECISION RATIO, FIBO(50),
     1 LBOUND, HBOUND, INTER, FINTER, DELTA, TESTLB,
     2 TESTHB, TLBV, THBV, TEST, FACT, TLB, F
        INTEGER EXPCNT, EXPNO, FLAG
        LBOUND      = 0.0
        TEST        = 1.0
        HBOUND      = 1.0
        FINTER      = 0.00001
        FACT        = 1.618034
C-----------------------------------------------------------------------
C       DETERMINE THE INTERVALS OF THE FIBONNACCI SEARCH
C-----------------------------------------------------------------------
 10     CONTINUE
        TLBV  = F( TEST )
        THBV = F( HBOUND )
        IF (TLBV.GT.THBV) GO TO 20
          TLB  = TEST
```

```
          TEST= HBOUND
          HBOUND = HBOUND * FACT
          GO TO 10
 20     CONTINUE
C-------------------------------------------------------------------
C       DETERMINE BOUNDS AND DELTA FOR FIBONNACCI SEARCH
C-------------------------------------------------------------------
        IF(TEST .NE. 1.) LBOUND = TLB
        INTER= HBOUND - LBOUND
        DELTA       = TEST - LBOUND
        TESTLB      = TEST
        TESTHB      = HBOUND - DELTA
        IF (TESTLB .LT. TESTHB) GOTO 38
        TLB = TESTLB
        TESTLB      = TESTHB
        TESTHB      = TLB
        DELTA       = TESTLB     - LBOUND
        TSTHB       = HBOUND     - DELTA
 38   CONTINUE
        INTER       = HBOUND     - LBOUND
        RATIO       = INTER/FINTER
C-------------------------------------------------------------------
C       DETERMINE THE NUMBER OF EXPERIMENTS REQUIRED TO HAVE
C       FINTER = 0.00001
C-------------------------------------------------------------------
        FIBO(1)     = 1
        FIBO(2)     = 1
        DO 39 I     = 3,50
          FIBO(I) = FIBO(I-1) + FIBO(I-2)
          IF (FIBO(I) .LT. RATIO) EXPNO = I + 1
 39     CONTINUE
C-------------------------------------------------------------------
C       START CLOSED BOUND FIBONNACCI SEARCH
C-------------------------------------------------------------------
        DO 40 EXPCNT=1, EXPNO
        TLBV= F(TESTLB)
        THBV= F(TESTHB)
        IF (TLBV.GE.THBV) GO TO 30
          LBOUND= TESTLB
          INTER= HBOUND - LBOUND
          DELTA= INTER - DELTA
          TESTLB= TESTHB
          TESTHB= HBOUND - DELTA
          FLAG  = 1
          GO TO 40
```

```
 30         CONTINUE
            HBOUND= TESTHB
            INTER= HBOUND - LBOUND
            DELTA= INTER - DELTA
            TESTHB= TESTLB
            TESTLB= LBOUND + DELTA
            FLAG = 0
 40         CONTINUE
          IF (FLAG .EQ. 1) FIBON = TESTLB
          IF (FLAG .EQ. 0) FIBON = TESTHB
          RETURN
          END
C-------------------------------------------------------------------
C         FUNCTION EVALUATION FOR FIBONNACCI SEARCH
C-------------------------------------------------------------------
          DOUBLE PRECISION FUNCTION F( K )
          DOUBLE PRECISION K, TEST(20), X(20), S(20)
          COMMON X, S, NTERM
          DO 10 I=1, NTERM
            TEST(I)= X(I) - K * S(I)
 10         CONTINUE
          F= -FUNCT( TEST )
          RETURN
          END
C-------------------------------------------------------------------
C         CALCULATION OF COST FUNCTION
C-------------------------------------------------------------------
          DOUBLE PRECISION FUNCTION FUNCT(X)
          DOUBLE PRECISION X(20)
          FUNCT=5.0*X(1)*X(1)+2.0*X(2)*X(2)+2.0*X(3)*X(3)
     &      +2.0*X(1)*X(2)+2.0*X(2)*X(3)
     &      -2.0*X(3)*X(1) -6.0*X(3)
          RETURN
          END
*********************************************************************
    INPUT DATA :
    NO. OF INDEPENDENT VARIABLES, NTERM  =   3
    STOPPING CRITERION, EPS              =   0.0001
    STARTING POINTS, X                   =   0.00   0.00   0.00

    RESULTS   :
    ITERATION       COST FUNCTION          VALUES OF X                K

         0                0.000            0.000   0.000          0.000   0.000
         1               -4.500            0.000   0.000          1.501   0.250
```

2	-7.500	1.000 -1.000	2.502 0.333
3	-9.000	1.000 -2.002	3.003 0.167

NORMAL TERMINATION OF THE BFGS PROGRAM

Program Description:

This program uses the Broyden, Fletcher, Goldfarb and Shanno (BFGS) algorithm to minimize an unconstrained multivariable function having as many as twenty variables. The program consists of a main program, two subroutines and three functions.

The three functions are as follows. The function FUNCT is the equation for the cost function to be minimized. The function F uses FUNCT for value of the cost function in the line search. The function FIBON uses the values of F in an open-ended Fibonacci line search. The two subroutines are SLOPE, which evaluates the partial derivatives using a forward difference approximation and PRINT, which prints the results of the computations.

The input data are the number of independent variables, starting point for the search and the stopping criterion, EPS. The program will terminate when the difference between the cost function values of two successive iterations is less than or equal to EPS, the stopping criteria.

The results are the iteration number, the values of the independent variables, and the cost function. Shown with the program are the input and output for the problem in Example 5-4.

The main program begins with an echo of the input data. Then it proceeds from iteration zero, the starting point, to use the BFGS algorithm to generate successive points until the stopping criterion is met. Initially, the Hessian matrix G is the identity matrix, and the gradient is computed using a forward difference approximation to the partial derivatives using subroutine SLOPE. The Fibonacci search function, FIBON, is used to locate the minimum along the gradient line from $\mathbf{x}_0$ to $\mathbf{x}_1$. Then the stopping criterion is checked, and the Hessian matrix **G** is updated. The value of the function is stored in ERROLD for future comparisons. The search direction to the next point is calculated and stored in the vector **S**. The value of the parameter of the line in the search direction, K, is calculated using FIBON to locate the next point. The value of the function at the new point is calculated and stored in ERR. The values of the iteration counter, the function at the new point, and the new point are printed using PRINT. The values of the gradient at the current point are computed and stored in the vector GRAD. The Hessian matrix **G** is updated, and the program returns to repeat the calculation until the error criterion is met.

To solve other problems, supply the equation to be minimized in the function FUNCT. It is used only by the procedure FIBON. If more than 20 variables are needed, then the CONST SIZE should be changed to the required number. No other modifications are needed. If this program is to be run in an 8- bit microcomputer, the real variables must be declared double precision to prevent underflow. Otherwise a division by zero will occur.

References

1. Simon, J. D. and H. M. Azma, "Exxon Experience with Large Scale Linear and Nonlinear Programming Applications", *Computers and Chemical Engineering*, Vol. 7, No. 5, p. 605 (1983).
2. Waren, A. D. and L. S. Lasdon, "The Status of Nonlinear Programming Software", *Operations Research*, Vol. 27, No. 3, p. 431 (May-June 1979).
3. Lasdon, L. S., "A Survey of Nonlinear Programming Algorithms and Software", *Foundations of Computer-Aided Chemical Process Design*, Vol. 1, p. 185, American Institute of Chemical Engineers, New York (1981).
4. Fletcher, Roger, *Practical Methods of Optimization, Vol. I, Unconstrained Optimization*, John Wiley and Sons, Inc., New York (1981).
5. Fletcher, Roger, *Practical Methods of Optimization, Vol. II, Constrained Optimization*, John Wiley and Sons, Inc., New York (1981).
6. Gill, P. E., E. Murray and M. H. Wright, *Practical Optimization*, Academic Press, New York (1981).
7. McCormick, G. P., *Nonlinear Programming: Theory, Algorithms and Applications*, John Wiley and Sons, Inc., New York (1983).
8. Himmelblau, D. M., *Applied Nonlinear Programming*, McGraw-Hill Book Company, New York (1972).
9. Avriel, Mordecai, *Nonlinear Programming: Methods and Analysis*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1976).
10. Wilde, D. J., *Optimum Seeking Methods*, Prentice-Hall Inc., Englewood Cliffs, New Jersey (1964).
11. Avriel, M., "Nonlinear Programming", Chapter 11 in *Mathematical Programming for Operations Researchers and Computer Scientists*, Ed. A. G. Holtzman, Marcel Dekker, Inc., New York (1981).
12. Wilde, D. J. and C. S. Beightler, *Foundations of Optimization*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1967).
13. Fletcher, Roger, op. cit. p. 57, Vol I.
14. Churchhouse, R. F., *Handbook of Applicable Mathematics, Vol. III, Numerical Methods*, John Wiley and Sons, Inc., New York (1981).
15. Reklaitis, G. V., A. Ravindran and K. M. Ragsdell, *Engineering Optimization, Methods and Applications*, John Wiley and Sons, Inc., New York (1983).
16. Kuester, J. L. and J. H. Mize, *Optimization Techniques with Fortran*, McGraw-Hill Book Company, New York (1973).
17. Smith, C. L., R. W. Pike and P. W. Murrill, *Formulation and Optimization of Mathematical Models*, International Textbook Company, Scranton, Pennsylvania (1970).
18. Griffith, R. E. and R. A. Stewart, " A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems", *Management Science*, Vol. 7, p. 379 (1961).
19. Lasdon, L. S., op. cit., p. 202.
20. Sandgren, Eric, *The Utility of Nonlinear Programming Algorithms*, Ph.D. dissertation, Purdue University, West Lafayette, Indiana (1977).

21. Colville, A. R., *A Comparative Study of Nonlinear Programming Codes*, IBM New York Scientific Center Report No. 320 - 2949, IBM Corporation, New York Scientific Center, 410 East 62nd Street, New York, NY 10021 (June 1968).
22. Lasdon, L. S. and A. D. Waren, "Large Scale Nonlinear Programming," *Computers and Chemical Engineering*, Vol. 7, No. 5, p. 595 (1983).
23. Franklin, Joel, *Methods of Mathematical Economies, Linear and Nonlinear Programming, Fixed Point Theorems*, Springer-Verlag Inc., New York (1980).
24. Vanderplaats, G. N., *Numerical Optimization Techniques for Engineering Design with Applications*, McGraw - Hill Book Company, New York (1984).
25. Hillier, F. S. and G. J. Lieberman, *Operations Research*, Holden - Day, Inc., San Francisco (1974).
26. Walsh, G. R., *Methods of Optimization*, John Wiley & Sons Inc., New York (1975).
27. McMillan, Jr., Claude, *Mathematical Programming: An Introduction to the Design and Application of Optimal Decision Machines*, John Wiley & Sons, Inc., New York (1970).
28. Gottfried, B. S. and Joel Weisman, *Introduction to Optimization Theory*, Prentice - Hall, Inc., Englewood Cliffs, New Jersey (1973).
29. Wolfe, P., "Methods of Nonlinear Programming" in *Recent Advances in Mathematical Programming*, Ed. R. L. Graves and P. Wolfe, McGraw - Hill Book Company, New York (1963).
30. Abadie, J. and J. Carpentier, "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints," in *Optimization*, Ed. R. Fletcher, Academic Press, London (1969).
31. Pollack, A. W. and W. D. Lieder, "Linking Process Simulators to a Refinery Linear Programming Model" in *Computer Applications to Chemical Engineering*, Ed. R. G. Squires and G. V. Reklaitis, ACS Symposium Series No. 124, American Chemical Society, Washington, D. C. (1980).
32. O'Neil, R. P., M. A. Williard, Bert Wilkins and R. W. Pike, "A Mathematical Programming Model for Natural Gas Allocation," *Operations Research*, Vol. 27, No. 5, p. 857 - 873 (Sept./Oct. 1979).
33. Sargent, R. W. H., "A Review of Optimization Methods for Nonlinear Problems" in *Computer Applications to Chemical Engineering*, Ed. R.G. Squires and G. V. Reklaitis, ACS Symposium Series No. 124, American Chemical Society, Washington, D.C. (1980).
34. Cooper, Leon, and David Steinberg, *Introduction to Methods of Optimization*, W. B. Saunders Company, Philadelphia (1970).
35. Adby, P. R. and M. A. H. Dempster, *Introduction to Optimization Methods*, John Wiley and Sons, Inc., New York (1974).
36. Bracken, J. and G. P. McCormick, *Selected Applications of Nonlinear Programming*, p. 16f, John Wiley and Sons, Inc. New York (1968).
37. Ray, W. H. and J. Szekely, *Process Optimization with Applications in Metallurgy and Chemical Engineering*, John Wiley and Sons, Inc., New York (1973).
38. April, G. C. and R. W. Pike, "Modeling Complex Chemical Reaction Systems," *Industrial and Engineering Chemistry, Process Design and Development*, Vol. 13, No. 1, p.1 (January 1974).

39. Fletcher, R., "Methods Related to Lagrange Functions," *Numerical Methods for Constrained Optimization*, Ed. P. E. Gill and W. Murray, Academic Press, New York (1974).
40. Doering, F. J. and J. L. Gaddy, "Optimization of the Sulfuric Acid Process with a Flowsheet Simulator," *Computers and Chemical Engineering*, Vol. 4, p. 113 (1980).
41. Martin, D. L. and J. L. Gaddy, "Modeling the Maleic Anhydrate Process," Summer National Meeting, American Institute of Chemical Engineers, Anaheim (May 20 - 24, 1984).
42. Jirapongphan, S., J.F. Boston, H. I. Britt and L. B. Evans, "A Nonlinear Simultaneous Modular Algorithm for Process Flowsheeting Optimization," American Institute of Chemical Engineers Annual Meeting, Chicago (November 1980).
43. Murtagh, B. A. and M. A. Saunders, *MINOS 5.0 Users Guide*, Technical Report SOL 83 - 20, Systems Optimization Laboratory, Department of Operations Research, Stanford University (December 1983).
44. Beigler, L. T. and R. R. Hughes, "Process Optimization: A Comparative Case Study," *Computers and Chemical Engineering*, Vol. 7, No. 5, p.645 (1983).
45. Locke, M. H. and A. W. Westerberg, "The ASCEND-II System - A Flowsheeting Application of a Successive Quadratic Programming Methodology," *Computers and Chemical Engineering*, Vol. 7, No. 5, p. 615 (1983).
46. Locke, M. H. , A. W. Westerberg, and R. H. Edahl, "Improved Successive Quadratic Programming Optimization Algorithm for Engineering Design Problems," *AIChE Journal,* Vol. 29, No. 5, p.871 (September, 1983).
47. Palacios-Gomez, F., L. Lasdon and M. Engquist, "Nonlinear Optimization by Successive Linear Programming," *Management Science*, Vol. 28, No. 5, p.871 (September 1983).
48. Chen, H. S. and M. A. Stadtherr, "Enhancements of the Han-Powell Method for Successive Quadratic Programming," *Computers and Chemical Engineering*, Vol. 8, No. 3/4, p. 229 (1984).
49. Biegler, L. T. and R. R. Hughes, "Infeasible Path Optimization with Sequential Modular Simulators," *AIChE Journal*, Vol. 28, No. 6, p. 994 (November 1982).
50. Bertsekas, Dimitri P., *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York (1982).
51. Han, S. P., "A Globally Convergent Method for Nonlinear Programming," *Journal of Optimization Theory and Applications*, Vol. 22, No. 3, p. 297 (July, 1977).
52. Han, S. P., "Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems," *Mathematical Programming*, Vol. 11, p. 263, Noth-Holland Publishing Company (1976).
53. Biegler, L. T. and J. E. Cuthrell, "Improved Infeasible Path Optimization for Sequential Modular Simulators - II: The Optimization Algorithm," *Computers and Chemical Engineering*, Vol. 9, No. 3, p. 257 (1985).
54. Haftka, R. T. and M. P. Kamat, *Elements of Structural Optimization*, Martinus Nijhoff Publisher, Dordrecht, The Netherlands (1985).
55. Dennis, J. E., and R. B. Schnable, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice - Hall Inc., Englewood Cliffs, New Jersey (1983).
56. Bazaraa, M. S. and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, John Wiley and Sons, Inc., New York (1979).

57. Powell, M. J. D., "An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives," *The Computer Journal*, Vol. 7, p. 155 (1964).
58. Drud, Arne, "CONOPT: A GRG Code for Large Sparce Dynamic Nonlinear Optimization Problems," *Mathematical Programming*, Vol. 31, p. 153 (1985).
59. Hadley, G. H., Nonlinear and Dynamic Programming, Addison - Wesley Publishing Company, Inc., Reading, Mass., p. 191 (1964).

Problems

5-1.[10] A Fibonacci search can be used to find the point on a line in space where a function is a maximum. For the two points (1, −1, 0, 2) and (−5, −1, 3, 1) use a Fibonacci search assuming perfect resolution and unimodality.

Give the coordinates of the points where the first two experiments would be placed assuming a total of five measurements will be used. What is the final interval of uncertainty on the coordinate axis x_1?

5-2.[10] In the following table eight values of y are given, and y is a function of four independent variables.

x_1	x_2	x_3	x_4	y
0	1	−1	3	5
1	1	−1	3	7
2	1	−1	3	9
−1	2	−1	3	2
0	−1	−1	3	7
0	1	1	3	7
0	1	−1	2	
0	2	0	3	5

a. Determine the line of steep ascent passing through the point (0, 1, −1, 3).
b. Determine the contour tangent hyperplane passing through (0, 1, −1, 3).

5-3[17] Use the method of gradient partan to find the minimum of the following function starting at (2, 1, 3).

$$y = x_1^2 + 3x_2^2 + 5x_3^2$$

5-4. For the following function draw contours on a graph for values of y of 20.0, 40.0, 60.0 and 80.0 in the region $0 \le x_1 \le 10$ and $0 \le x_2 \le 10$.

$$y = x_1 x_2$$

Starting at point $\mathbf{x}_0(4, 4)$ apply Pattern Search to move toward maximum and employ a step size $\boldsymbol{\delta}(1/2, 1/2)$. Make local explorations and accelerations (pattern moves) to obtain the points through $\mathbf{b}_5$.

5-5. In Figure 5-12 a contour map is given for a function with a maximum located in the upper center. For the four multivariable search techniques, gradient search, sectioning, gradient partan and pattern search, sketch (precisely) the path these algorithms would take, beginning at the indicated starting point and going toward the maximum. For pattern search make the step-size equal to one-half of the width of the grid. The pattern search step size can be cut in half for the search to continue, if necessary. This will be the resolution limit, however. In addition, make brief comments about the effectiveness of these four techniques as applied to this function.

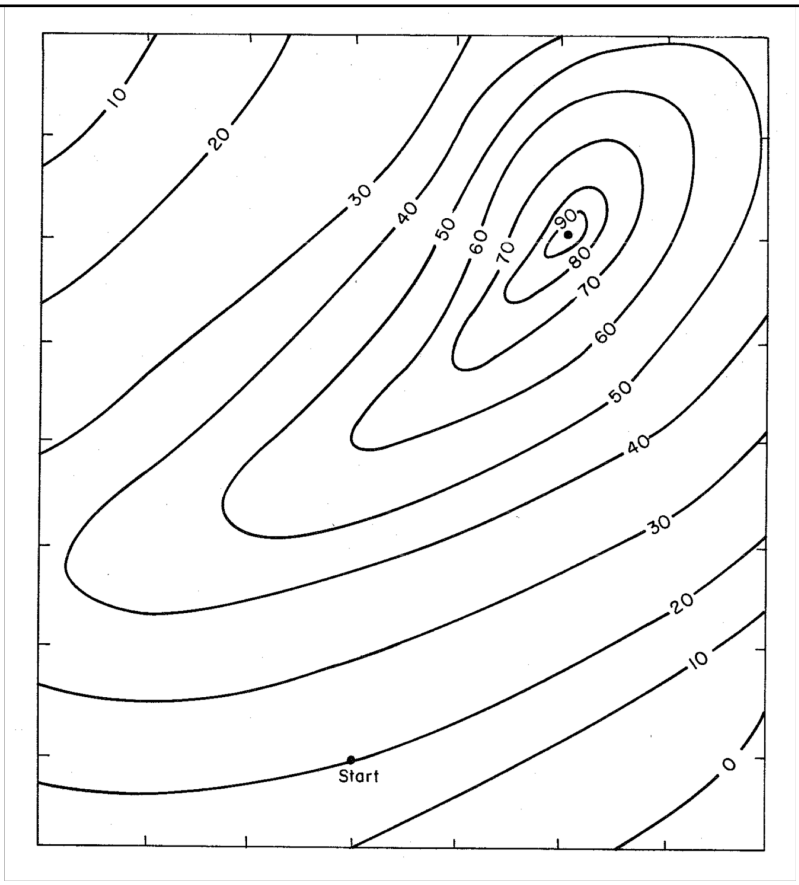


Figure 5-12 Contour Map of a Function with a Maximum Located in the Upper Center for Problem 5-5

5-6. On the contour map given in Figure 5-13, sketch (precisely) the path of gradient partan, Powell's method and pattern search beginning at the starting point shown. For pattern search have the step size initially equal to the grid shown on the contour map and reduce the step size by one-half to have the search continue. Reduce the step size by one-half again if necessary, to have

pattern search continue. Give a brief discussion of the performance of these methods on this function.

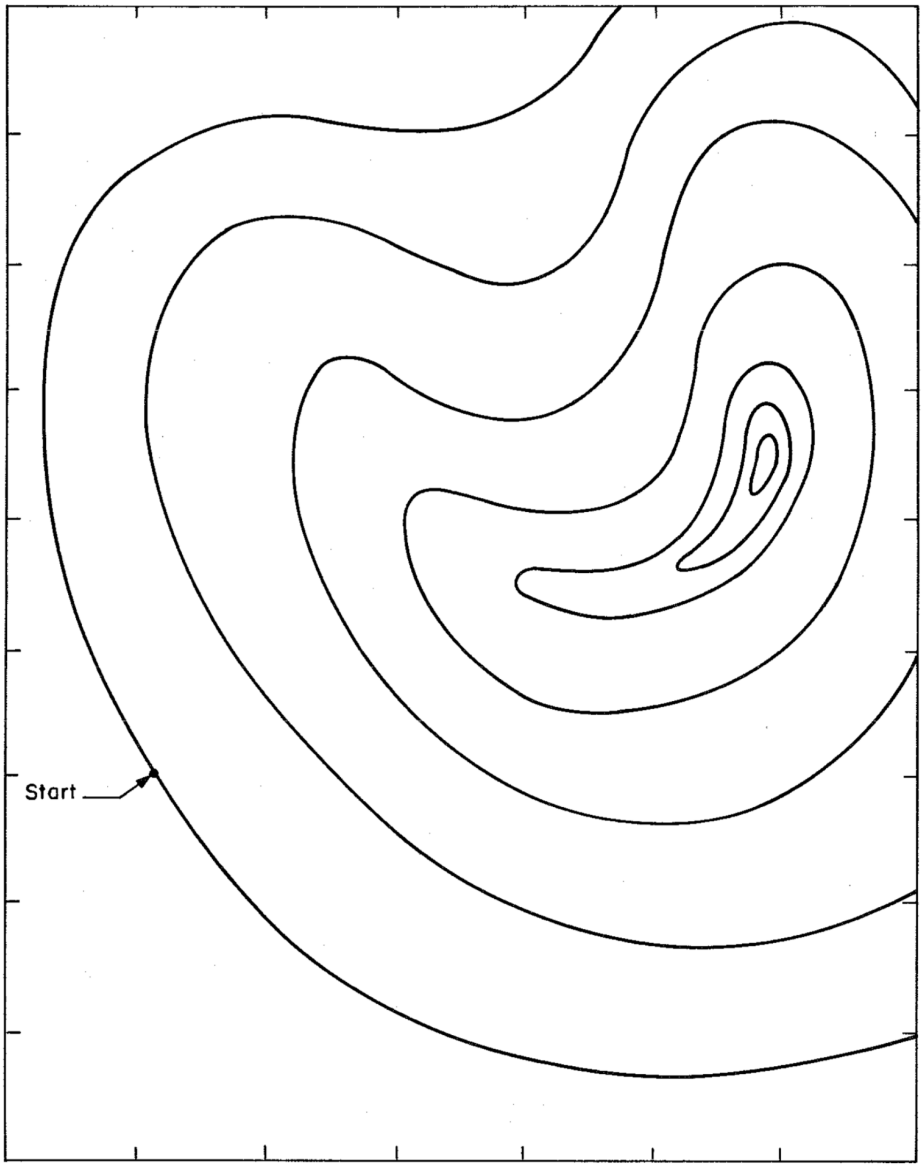


Figure 5-13 Contour Map for Problem 5-6

5-7. Newton's method is obtained from the Taylor series expansion for $y(\mathbf{x})$, truncating the terms which are third order and higher, Equation 5-8. Then Equation 5-12 is obtained from the quadratic approximation, where $\mathbf{x}$ is the location of the minimum of the quadratic approximation. Discuss the iterative procedure that would be used to move to an optimum. To ensure convergence to a

minimum (maximum), the value of $dy(\alpha)/d\alpha$ always must be negative (positive), where α is the parameter of the line between points $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ obtained from successive applications of the algorithm.

$$\mathbf{x} = \mathbf{x}_k + \alpha\,(\mathbf{x}_{k+1} - \mathbf{x}_k)$$

Explain why this restriction is required for convergence to a local minimum (maximum).

5-8. Search for the minimum of the following function using gradient search starting at point $\mathbf{x}_0$ (1, 1, 1).

$$y = x_1^2 + x_2^2 + x_3^2$$

5-9. Develop and use a simplified version of Newton's method (quadratic fit) to search for the minimum of the function given in Problem 5-8 starting at the same point. Give the Taylor series expansion for three independent variables truncating third and higher order terms neglecting interacting (mixed partial derivative) terms for simplicity. Differentiate the truncated Taylor series equation of with respect to x_1, x_2 and x_3 to compute the optimum of the quadratic approximation, x_1^*, x_2^* and x_3^*. Then apply these results to minimize the function of the problem. Compare the effort required for one iteration of the linear algorithm in Problem 6-8 to one iteration of the quadratic algorithm.

5-10. In Problem 6-7 a simplified alkylation process with three identical reactors in series is described. The profit function for each reactor can be represented by an equation with elliptical contours, and the catalyst degradation function can be represented by a linear equation.

a. If the optimum of the profit function for an individual reactor is at $F = 10$ and $C = 95$, derive the profit function to be maximized and the constraint equations to be satisfied for the process. The profit function for one reactor is given by the following equation.

$$y = 150 - 6(F - 10)^2 - 24(\boldsymbol{C} - 95)^2$$

The constraint equations have the form $y = mx + b$, and the parameters m and b can be determined from Figure 6-32.

b. Form the penalty function for the above problem and discuss how this form will maximize the profit function and satisfy the constraint equations when a search technique is used to find the optimum.

5-11. Solve the following optimization problem by successive linear programming starting at $\mathbf{x}_0$ (0, 1/2) using limits of (1, 1). Reduce the limits by one-half if infeasible points are encountered.

$$\text{minimize:} \qquad (x_1 - 2)^2 + (x_2 - 1)^2$$

subject to: $(-1/4)\, x_1 - x_2^2 + 1 \geq 0$

$x_1 - 2x_2 + 1 = 0$

5-12. Solve the following optimization problem by successive linear programming starting at point $\mathbf{x}_0$ (1, 1) using limits of (1, 1). Reduce the limits by one-half if infeasible points are encountered.

maximize: $4x_1 + x_2$

subject to: $x_1^2 + 2x_2^2 \leq 20.25$

$x_1^2 - x_2^2 \leq 8.25$

5-13. Solve the following optimization problem by successive linear programming starting at point $\mathbf{x}_0$ (1, 1) using limits of (1, 1). Reduce the limits by one-half if infeasible points are encountered.

minimize: $y = 2x_1^2 + 2x_1x_2 + x_2^2 - 20x_1 - 14x_2$

subject to: $x_1^2 + x_2^2 \leq 25$

$x_1^2 - x_2^2 \leq 7$

5-14.[26] Solve the following problem by successive linear programming starting at point (2, 1) using limits of (1/2, 1/2). Reduce the limits by one-half if infeasible points are encountered.

maximize: $2x_1^2 - x_1x_2 + 3x_2^2$

subject to: $3x_1 + 4x_2 \leq 12$

$x_1^2 - x_2^2 \geq 1$

5-15.[34] Solve the following problem by successive linear programming starting at point $\mathbf{x}_0$ (1, 1) using limits of (2, 2). Reduce the limits by one-half if infeasible points are encountered.

maximize: $3x_1^2 + 2x_2^2$

subject to: $x_1^2 + x_2^2 \leq 25$

$9x_1 - x_2^2 \leq 27$

5-16. The following multivariable optimization problem is shown in Figure 5-7.

minimize: $-2x_1 - 4x_2 + x_1^2 + x_2^2 + 5$

subject to: $-x_1 + 2x_2 \leq 2$

$x_1 + x_2 \leq 4$

a. Give the successive linear programming algorithm for this problem in the form of Equations 6-34. The upper and lower bounds are the same and are equal to 1.0.

b. For starting point $\mathbf{x}_0 = (0,0)$ apply the algorithm from part a to search for the optimum by successive linear programming.

5-17. Solve Problem 5-16 by quadratic programming.

5-18.[26] Solve the following problem by quadratic programming.

maximize: $-2x_1^2 - x_2^2 + 4x_1 + 6x_2$

subject to: $x_1 + 3x_2 \leq 3$

5-19.[27] Solve the following problem by quadratic programming.

maximize: $6x_1 - 2x_1^2 + 2x_1x_2 - 2x_2^2$

subject to: $x_1 + x_2 \leq 2$

5-20.[28] Solve the following problem by quadratic programming.

maximize: $9x_2 + x_1^2$

subject to: $x_1 + 2x_2 = 10$

5-21. Solve the following problem by quadratic programming.

minimize: $2x_1^2 + 2x_1x_2 + x_2^2 - 20x_1 - 14x_2$

subject to: $x_1 + 3x_2 \leq 5$

$2x_1 - x_2 \leq 4$

5-22. Solve the following problem by the generalized reduced gradient method starting at the feasible point $\mathbf{x}_0(1, 1, 19)$ to find the optimum located at $\mathbf{x}^*(4, 3, 0)$. Use the optimum point to determine the appropriate value of the parameter of the reduced gradient line for one line search to arrive at the optimum.

maximize: $3x_1^2 + 2x_2^2 - x_3$

subject to: $x_1^2 + x_2^2 = 25$

$9x_1 - x_2^2 + x_3 = 27$

5-23.[11] Solve the following problem by the generalized reduced gradient method. Start at the point $\mathbf{x}_0 = (2, 1, 3, 1)$ and have x_1 and x_4 be the basic or dependent variables and x_2 and x_3 the nonbasic or independent variables.

minimize: $x_1^2 + 4x_2^2$

subject to: $x_1 + 2x_2 - x_3 = 1$

$-x_1 + x_2 + x_4 = 0$

5-24. Solve the following problem by the generalized reduced gradient method starting at point $\mathbf{x}_0\,(2, 4, 5)$. Show that the value of the parameters of the reduced gradient line $\alpha_1 = -1/20$ locates the minimum of the economic model and satisfies the constraints.

minimize : $4x_1 - x_2^2 + x_3^2 - 12$

subject to: $-x_1^2 - x_2^2 + 20 = 0$

$x_1 + x_3 - 7 = 0$

5-25.[17] Find the minimum of the following function starting at the point $\mathbf{x}_0\,(1, 1, 1)$. However, this time experimental error is involved; and the Kiefer-Wolfowitz procedure must be used, employing $a_k = 1/k$ and $c_k = 1/k^{1/4}$ with $k = 1, 2, ..., 12$. Simulate experimental error by flipping a coin and adding (subtracting) 0.1 from y if the coin turns up heads (tails).

$$y = x_1^2 + 3x_2^2 + 5x_3^2$$

5-26. Solve the following problem by successive quadratic programming and the generalized reduced gradient method, starting at point $\mathbf{x}_0\,(0, 1/2)$, and compare these results to the solution given in Figure 6-8.

minimize: $(x_1\text{-}2)^2 + (x_2\text{-}1)^2$
subject to: $-\frac{1}{4}\,x_1^2 - x_2^2 + 1 \geq 0$
$x_1 - 2x_2 + 1 = 0$