THE CONJUGATE GRADIENT METHOD AND MANAGEMENT SYSTEMS

by

CHRISTOPHER REBELO NUNES

B.E.(Elec.)(Hons.), B.E.(Mech.)(Hons.), University of Bombay
Bombay, India, 1966, 1967

A MASTER'S REPORT

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1969

Approved by

Major Professor

TABLE OF CONTENTS

1. INTRODUCTION

Linear programming has long been used to solve linear problems of management systems, but no effective algorithms have been developed for solving a general nonlinear programming problem. Several methods like the calculus of variation, dynamic programming and maximum principle have been recently developed. However, none of these techniques are very useful for solving complex management systems. The dynamic programming technique [12] faces dimensionality difficulty, whereby it cannot be used to solve problems with a large number of state variables. This is a severe limitation. Although several techniques are available to overcome this difficulty, they essentially trade computer memory for computer time. None of these techniques are as efficient as the original dynamic programming algorithm.

It is desirable to distinguish between two classes of nonlinear programming problems. The first class consists of nonlinear objective function and linear constraints while the second has a linear or nonlinear objective function and nonlinear constraints. The former is the less difficult of the two. In developing techniques for solving this class of problems, much can be gained from a knowledge of the established methods for unconstrained optimization [17]. Along these lines two possible approaches can be followed. One is to transform the constrained problem into unconstrained maximization problems, followed by the use of known techniques to solve each maximization problem. This type of approach is illustrated by Rosenbrock [16]. The other theoretically more satisfying approach, is to modify and extend methods of unconstrained maximization so that they can handle inequality constraints directly. The best example of this is Rosen's gradient projection method [15], which is an extension of the steepest ascent technique.

Unfortunately it suffers the same drawbacks i.e. slow rate of convergence for a nonlinear function. The variable metric method developed by Davidon [4] and later simplified by Fletcher and Powell [5] is a most powerful quadratically convergent technique for unconstrained optimization. Extension of the work of Davidon and Fletcher and Powell to the case of maximization under linear constraints appeared to be a promising path of attack in the solution of management problems.

The conjugate gradient method of Goldfarb [7,8,9] shows a significant improvement over existing nonlinear programming techniques in the rate of convergence. The purpose of this report is to apply the above method to industrial management problems characterized by linear constraints. This method is in many ways analogous to Rosen's gradient projection technique.

The geometrical interpretation and the outline of the algorithm is given. Two management problems are solved using this algorithm, and their optimum results are tabulated. The computational aspects with the aid of flow charts and computer programs are described in detail.

2. THE CONJUGATE GRADIENT METHOD

This method is based upon Davidon's variable metric method for unconstrained minimization. The conjugate gradient technique starts with an initial point in the m-dimensional variable space. If the given initial point is not feasible, a feasible one (satisfying all constraints) is obtained using Rosen's technique [15] for locating a feasible point. Starting with this feasible point a stepwise procedure is followed which gives a new feasible point at each step with an increased value of the objective function for a maximization problem. As the method proceeds, information about the local curvature of the objective function is incorporated into a matrix $\underline{H}_q^i$.

The length of the step is taken as the maximum possible without leaving the feasible region i.e. violating the constraint. Directions of search are determined by premultiplying the vector gradient $\underline{g}$ by the matrix $\underline{H}_q^i$. This variable metric $\underline{H}_q^i$ is modified when a constraint is added to or removed from the basis. To locate the maximum a cubic interpolation scheme of Davidon [4] given in the Appendix B is used.

2.1 Formulation of the problem

The conjugate gradient algorithm given below is used to obtain a local maximum of a nonlinear objective function

$$f(\underline{x}) = f(x_1, x_2, \ldots\ldots, x_m) \tag{1}$$

of m variables x_j, j=1,2, ..., m subject to linear equality and inequality constraints of the type

$$\sum_{j=1}^{m} a_{ij} x_j - b_i = 0, \quad i = 1,2, \ldots\ldots, e$$

$$\sum_{j=1}^{m} a_{ij} x_j - b_i \geq 0, \quad i=e+1,e+2,.....,k$$

This formulation includes as a special case non-negativity constraints (bounds) common in both linear and nonlinear programming problems.

On normalizing the above constraints we have

$$\sum_{j=1}^{m} n_{ij} x_j - b_i = 0, \quad i=1,2,.....,e \tag{2}$$

$$\sum_{j=1}^{m} n_{ij} x_j - b_i \geq 0, \quad i=e+1,e+2,.....,k \tag{3}$$

where the n_{ij} have been normalized such that

$$\sum_{j=1}^{m} (n_{ij})^2 = 1, \quad i=1,2,.....,k \tag{14}$$

It is obvious that bounds need not be normalized.

Geometrically x_i, i=1,2,.....,m represents a point in an m-dimensional Euclidean space E^m. Such a point can be represented by an m-dimensional column vector

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

The transpose of the vector $\underline{x}$ is denoted by $\underline{x}^T$. Subscripts on vectors

usually indicate components while superscripts in general differentiate between points.

Let $\underline{n}_i$ denote the column vector

$$\underline{n}_i = \begin{bmatrix} n_{i1} \\ n_{i2} \\ \vdots \\ n_{im} \end{bmatrix}$$

Then equations (2) through (4) can be written as

$$\underline{n}_i^T \underline{x} - b_i' = 0, \quad i=1,2,.....,e \tag{5}$$

$$\underline{n}_i^T \underline{x} - b_i' \geq 0, \quad i=e+1,e+2,....,k \tag{6}$$

where the $\underline{n}_i$ are unit normals, such that

$$\underline{n}_i^T \underline{n}_i = 1, \quad i=1,2,.....,k \tag{7}$$

The constraints (5) and (6) represent e hyperplanes and (k-e) closed half spaces, whose intersection in E^m is a convex polyhedral region R often called the "feasible" region. This region is assumed to be bounded and for this to be true it is necessary that $k \geq m+1$. Any point x such that $x \in R$ is called a "feasible" point. The hyperplane corresponding to a strict equality for a particular i in equation (6) is called the "defining" hyperplane for the associated half space. The boundary B of R is the intersection of R with the union of the (k-e) defining hyperplanes.

A set of hyperplanes is linearly independent if the set of unit normals to these hyperplanes, is linearly independent. If q are the number of linearly independent defining hyperplanes, then the intersection of these hyperplanes forms an (m-q) dimensional subspace M_q of E^m. If movement is restricted to M_q, then these hyperplanes are referred to as the "constraint basis". $\tilde{M}_q$ is a q-dimensional subspace spanned by the q unit normals to these q hyperplanes, hence $\tilde{M}_q$ is the orthogonal component to M_q i.e.

$$M_q \cap \tilde{M}_q = 0$$

and

$$M_q \cup \tilde{M}_q = E^m$$

2.2 Geometrical interpretation

It is convenient to use geometrical concepts to describe the maximization procedure. As previously mentioned $\underline{x}^i$ is a point in the m-dimensional Euclidean space E^m. The gradient at this point is given by

$$g(\underline{x}^i) = \left\{\frac{\partial f(\underline{x}^i)}{\partial x_j}\right\}$$

The components of this gradient can be considered as the coordinates of a point in another m-dimensional Euclidean space. If $f(\underline{x})$ is differentiable at all points $\underline{x}^i$, then each point $\underline{x}^i$ in the position space has associated with it a point $g(\underline{x}^i)$ in the gradient space. In addition, if $f(\underline{x})$ is twice differentiable, then in the neighborhood of any point $\underline{x}^i$ in E^m the second partial derivatives of $f(\underline{x})$ specify a linear mapping of changes in position, $\underline{dx}$, onto changes in gradient $\underline{dg}$, according to the relation

$$\underline{dg} = G(\underline{x})\ \underline{dx}$$

where the Hessian matrix $G(\underline{x})$ of second partial derivatives is given by

$$G(\underline{x}) = \left\{\frac{\partial^2 f(\underline{x})}{\partial x_i \partial x_j}\right\}$$

The vectors $\underline{dx}$ and $\underline{dg}$ will be directed likewise if $\underline{dx}$ is an eigenvector of $G(\underline{x})$. If the ratios among the corresponding eigenvalues of the matrix $G(\underline{x})$ are large, then for most $\underline{dx}$ there will be a considerable difference in the directions of $\underline{dx}$ and $\underline{dg}$.

Consider the problem of locating the point at which a strictly concave quadratic function

$$f(\underline{x}) = f_0 + \underline{a}^T \underline{x} + \frac{1}{2} \underline{x}^T \underline{G} \underline{x} \tag{8}$$

is maximized. Then the Hessian matrix is equal to the constant negative definite matrix $\underline{G}$ over the entire space E^m and we have

$$\underline{g}^{i+1} - \underline{g}^i = \underline{G}(\underline{x}^{i+1} - \underline{x}^i) \tag{9}$$

where $\underline{g}^i = g(\underline{x}^i)$

Therefore $$\underline{x}^{i+1} - \underline{x}^i = \underline{G}^{-1} (\underline{g}^{i+1} - \underline{g}^i) \tag{10}$$

where $\underline{G}^{-1}$ is the inverse of $\underline{G}$.

In the unconstrained case, where the feasible region R is all of E^m, a necessary and sufficient condition for a global maximum is that $g(\underline{x}) = 0$. The point at which this maximum occurs can then be found in one step by computing

$$\underline{x}^{i+1} = \underline{x}^i - \underline{G}^{-1} \underline{g}^i \tag{11}$$

This, of course, is just Newton's method (referred to as the second-order gradient method), for solving the set of m equations in m unknowns,

$g_j\ (x_1, x_2, \ldots\ldots, x_m) = 0$, $j = 1,2, \ldots\ldots, m$, resulting from the necessary and sufficient conditions for a global maximum of a strictly concave function.

It is obvious from the above discussion that the method of steepest ascent (i.e. moving in the gradient direction), and the iterative procedure based upon moving in a direction $-\underline{G}^{-1}\ \underline{g}^{i}$ can lead to very different paths to a solution.

In the above discussion it was assumed that the objective function $f(\underline{x})$ was quadratic in $\underline{x}$. If this were true or nearly so, then Newton's method might be the optimal maximization scheme to use. However, in most of the management problems $f(\underline{x})$ is not a quadratic function and this assumption is reasonably accurate only in a small neighborhood of a point. Therefore, evaluation of the Hessian matrix and inversion of it at each step might not, therefore, be the best policy to adopt as this can be computationally quite time consuming. In addition a serious drawback of second-order gradient methods is that for non-concave functions, convergence cannot be assured for starting points not close to the maximum.

Davidon's variable metric method [4] and its reformulation by Fletcher and Powell [5], takes a more satisfactory approach for locating the maximum of an unconstrained nonlinear function. Their method is to initially choose an approximation to $\underline{G}^{-1}$, and by using the actual relationship that exists between changes in $\underline{g}$ and in $\underline{x}$ improve this approximation with each subsequent iteration. This eventually results in a technique that incorporates the advantages of both Newton's and the steepest ascent methods, while avoiding their major limitations. As in the former method, Davidon's technique is stable and generates directions of search that are always

uphill, and does not require the computation of second partial derivatives. As in the latter method, it is quadratically convergent.

Consider now the problem of locating the point at which $f(\underline{x})$ assumes its maximum in M_q starting from a point $\underline{x}^i$. A necessary condition for $f^*(\underline{x})$ to be the global maximum of $f(\underline{x})$ for $\underline{x}$ in M_q is that the gradient $g^*(\underline{x})$ should lie in the subspace orthogonal to M_q i.e. lie in $\tilde{M}_q$. Therefore,

$$g^*(\underline{x}) = \underline{N}_q \underline{\alpha}$$

where $\underline{N}_q = [\underline{n}_1, \underline{n}_2, \ldots\ldots\ldots, \underline{n}_q]$ is an (m x q) matrix and $\underline{\alpha}$ is a q-dimensional column vector. The columns of $\underline{N}_q$ are the unit normals to the hyperplanes under consideration and lie in $\tilde{M}_q$. Substituting $\underline{g}^{i+1} = \underline{N}_q \underline{\alpha}$ into equation (10) we have

$$\underline{x}^{i+1} - \underline{x}^i = \underline{G}^{-1} (\underline{N}_q \underline{\alpha} - \underline{g}^i) \tag{12}$$

$(\underline{x}^{i+1} - \underline{x}^i)$ lies in subspace M_q. Therefore,

$$\underline{N}_q{}^T (\underline{x}^{i+1} - \underline{x}^i) = 0 = \underline{N}_q{}^T \underline{G}^{-1} \underline{N}_q \underline{\alpha} - \underline{N}_q{}^T \underline{G}^{-1} \underline{g}^i$$

Since the rank of $\underline{N}_q$ is $q \leq m$ and that of $\underline{G}^{-1}$ is $m \geq q$ the inverse of $(\underline{N}_q{}^T \underline{G}^{-1} \underline{N}_q)$ exists [3] and $\underline{\alpha}$ is given by

$$\underline{\alpha} = (\underline{N}_q{}^T \underline{G}^{-1} \underline{N}_q)^{-1} \underline{N}_q{}^T \underline{G}^{-1} \underline{g}^i$$

Equation (12) can then be written as

$$\underline{x}^{i+1} - \underline{x}^i = \underline{G}^{-1} (\underline{N}_q{}^T (\underline{N}_q{}^T \underline{G}^{-1} \underline{N}_q)^{-1} \underline{N}_q{}^T \underline{G}^{-1} \underline{g}^i - \underline{g}^i)$$

Therefore,

$$\underline{x}^{i+1} = \underline{x}^{i} + (\underline{G}^{-1}\,\underline{N}_q\,(\underline{N}_q^{T}\,\underline{G}^{-1}\,\underline{N}_q)^{-1}\,\underline{N}_q^{T}\,\underline{G}^{-1} - \underline{G}^{-1})\underline{g}^{i} \tag{13}$$

Equation (13) can be expressed as

$$\underline{x}^{i+1} = \underline{x}^{i} - \hat{\underline{P}}_q\,\underline{G}^{-1}\underline{g}^{i} \tag{14}$$

where $\hat{\underline{P}}_q = \underline{I} - \underline{G}^{-1}\,\underline{N}_q(\underline{N}_q^{T}\,\underline{G}^{-1}\,\underline{N}_q)^{-1}\,\underline{N}_q^{T}$

is a non-euclidean projection operator that projects E^m onto subspace M_q.

Equation (14) shows that to locate a maximum one should move in a direction $-\hat{\underline{P}}_q\,\underline{G}^{-1}\,\underline{g}^{i}$. As in the unconstrained case, a better approach for the nonlinear problem is one based upon Davidon's method. In the conjugate gradient algorithm described in the next section a trial value for the operator $-\hat{\underline{P}}_q\,\underline{G}^{-1}$ at a point in M_q is initially specified. This is the positive definite variable metric $\underline{H}_q$.

3. OUTLINE OF THE CONJUGATE GRADIENT ALGORITHM

This algorithm is essentially an extension of Fletcher and Powell's version of Davidon's variable metric method to the maximization of an objective function subject to linear constraints. It closely follows Rosen's gradient projection method; the main difference is that the directions of search are determined by premultiplying the gradient vector by the matrix $\underline{H}_q$ rather than by the orthogonal projection matrix $\underline{P}_q$. The algorithm is based upon the manner in which the matrix $\underline{H}_q$ is updated. In the course of the statement of this algorithm, eqs. (2) and (4) are given for modifying $\underline{H}_q$ when a hyperplane is either dropped from or added to the constraint basis, while equation (5), due to Davidon [4] is used to improve the approximation of $\underline{H}_q$ to $-\hat{\underline{P}}_q\, G^{-1}$. Recursion relations for obtaining the inverse matrix $(\underline{N}_q^T\, \underline{N}_q)^{-1}$ on changing the constraint basis, are given in the Appendix A.

Let $\underline{x}^i$ denote the current point and let $f(\underline{x}^i)$ and $g(\underline{x}^i)$ the corresponding value of the function and the gradient. The matrix operator $\underline{H}_q$ is therefore $\underline{H}_q{}^i$. The basic conjugate gradient algorithm put forward by Goldfarb [7] can be stated as follows:

(0) Initially some feasible point $\underline{x}^0$ is obtained and $\underline{H}_0{}^0$ is chosen to be any positive definite matrix, usually $\underline{I}$, the identity matrix. If the point $\underline{x}^0$ lies in the intersection M_q of q linearly independent hyperplanes, then these constraints are added to the constraint basis and $\underline{H}_q{}^0$ is computed by recursively using eq. (4) (given in step (iv) of this algorithm) q-times once for each hyperplane. The e hyperplanes that correspond to the equality constraints (eq. (5) in section 2.1), are added first. Compute the gradient $\underline{g}^0 = g(\underline{x}^0)$.

(i) Compute $\underline{\alpha}$ as follows

$$\underline{\alpha} = (\underline{N}_q^T \underline{N}_q)^{-1} \underline{N}_q^T \underline{g}^i \tag{1}$$

If $||\underline{g}^i|| = 0$ and $\alpha_j \leq 0$, $j = e+1, e+2, \ldots\ldots, q$ then

$\underline{x}^i$ is a global maximum. If this fails, go to (ii).

(ii) Compute $||\underline{H}_q^i \underline{g}^i||$ then either $2||\underline{H}_q^i \underline{g}^i|| > \max (0, \alpha_q d_{qq}^{-\frac{1}{2}})$ or $2||\underline{H}_q^i \underline{g}^i|| \leq \alpha_q d_{qq}^{-\frac{1}{2}}$, where d_{qq} is the qth diagonal element of the inverse matrix $(\underline{N}_q^T \underline{N}_q)^{-1}$ and where it is assumed that $\alpha_q d_{qq}^{-\frac{1}{2}} \geq \alpha_i d_{ii}^{-\frac{1}{2}}$, $i = e+1, e+2, \ldots\ldots, q-1$

If the former applies proceed to (iii). If the latter holds good, drop the qth hyperplane from the constraint basis and compute $\underline{H}_{q-1}^i$ from

$$\underline{H}_{q-1}^i = \underline{H}_q^i + \frac{\underline{P}_{q-1} \underline{n}_q \underline{n}_q^T \underline{P}_{q-1}}{||\underline{P}_{q-1} \underline{n}_q||^2} \tag{2}$$

where $$\underline{P}_{q-1} = \underline{I} - \underline{N}_{q-1} (\underline{N}_q^T \underline{N}_{q-1})^{-1} \underline{N}_{q-1}^T$$

Let $q = q-1$ and return to (i).

(iii) Let $\underline{s}^i = \underline{H}_q^i \underline{g}^i$ and compute λ^i from the following formulas:

$$\lambda_j = -(\underline{n}_j^T \underline{x}^i - b_j)/ \underline{n}_j^T \underline{s}^i, \quad j = q+1, q+2, \ldots\ldots, k$$

$$\lambda^i = \min (\lambda_j > 0) \tag{3}$$

Using the scheme given in the Appendix B, interpolate cubically to

obtain γ^i, $0 \leq \gamma^i \leq \lambda^i$ such that $f(\underline{x}^i + \gamma^i \underline{s}^i)$ is maximized. Set $\underline{x}^{i+1} = \underline{x}^i + \gamma^i \underline{s}^i$ and compute $\underline{g}^{i+1} = g(\underline{x}^{i+1})$

(iv) If $\gamma^i = \lambda^i$, add to the constraint basis the hyperplane corresponding to the min (λ_j) in (iii). Obtain $\underset{\sim}{H}_{q+1}^{i+1}$ from the relation

$$\underline{H}_{q+1}^{i+1} = \underline{H}_q^i - \frac{\underline{H}_q^i \underline{n}_j^T \underline{n}_j \underline{H}_q^i}{\underline{n}_j^T \underline{H}_q^i \underline{n}_j} \tag{4}$$

Set $q = q+1$ and $i = i+1$ and return to (i).

(v) Otherwise set $\underline{y}^i = \underline{g}^{i+1} - \underline{g}^i$ and $\underline{\sigma}^i = \gamma^i \underline{s}^i$ and update $\underline{H}_q^i$ as follows:

$$\underline{H}_q^{i+1} = \underline{H}_q^i + \underline{A}^i + \underline{B}^i \tag{5}$$

where $$\underline{A}^i = - \frac{\underline{\sigma}^i (\underline{\sigma}^i)^T}{(\underline{\sigma}^i)^T \underline{y}^i}$$

and $$\underline{B}^i = - \frac{\underline{H}_q^i \underline{y}^i (\underline{y}^i)^T \underline{H}_q^i}{(\underline{y}^i)^T \underline{H}_q^i \underline{y}^i}$$

The constraint basis, and consequently M_q remains unchanged. Set i = i+1 and return to (i).

The above equation (5) serves two purposes. The term $\underline{A}^i$ improves the approximation of $\underline{H}_q^i$ to $-\hat{\underline{P}}_q \underline{G}^{-1}$ and the term $\underline{B}^i$ generates mutually conjugate direction of search.

4. FUNCTIONAL DESCRIPTION OF THE COMPUTER LOGIC

In formulating a problem to be run on the IBM 360/50 the following quantities had to be defined for the successful operation of the conjugate gradient technique:

a) A set of variables, x_j, j = 1,2, ..., m

b) Linear constraints, C_i, i = 1,2, ..., k

c) The objective function, $f(\underline{x})$, to be maximized and its gradient $g(\underline{x})$.

d) A initial point $\underline{x}^0$.

Description of the constraints

The linear constraints may consist of inequalities or equalities or both. An inequality constraint, C_i, must be written in the form $\sum_{j=1}^{j=m} a_{ij} x_j \geq b_i$. The constraints form a convex region which is usually bounded. The maximum of a concave nonlinear function will be found even if the region is unbounded. However, the use of lower and upper bounds on some or all of the variables will often speed up the convergence to the maximum. For the technique to operate, at least one constraint is essential.

Constraints on only one variable are called bounds. Any constraint which can be expressed as $\pm\ x_j \geq b_i$ is considered as a bound. The input consists first of bounds, if any, which are numbered as $C_1, C_2, \ldots\ldots, C_b$, say, and then the constraints other than bounds, numbered as $C_{b+1}, C_{b+2}, \ldots\ldots, C_k$. Non-negativity constraints are not assumed by the program, and have to be expressed as bounds. A bound requires less computation time and should be used in place of a constraint, wherever possible.

The constraints C_i are considered as equality if they can be expressed

in the form $\sum_{j=1}^{m} a_{ij} x_j - b_i = 0$. The objective function $f(\underline{x})$ to be maximized and its gradient $g(\underline{x})$ are computed by the subroutine FUNCT. The program will always find a global maximum if $f(\underline{x})$ is a concave function (a linear function is a special case of a concave function) and has continuous gradient. However, solutions may be obtained even if the above two conditions are not satisfied. Generally, if $f(\underline{x})$ is not concave, a local maximum will be found which may depend on the initial point chosen. Under such circumstances several widely seperated starting points should be tried. If different results are obtained, the best that can be done is to take the maximum value of this set of local maxima. A minimization problem can be solved by this algorithm as maximizing the negative of the original function, namely maximizing $-f(\underline{x})$.

The initial point $\underline{x}^0$ need not be feasible, as the program will first obtain a feasible point, if one exists. However, this procedure can be time consuming. The best estimate of the feasible point is preferred. The closer the starting point is to the optimum, the lesser is the time consumed for obtaining the correct solution.

As input, m number of variables, k total number of constraints including bounds, b number of bounds, and e number of equality constraints, are read first. If $b \neq 0$, the b subscripts for the bounds are read next. these are $\pm$ j's, representing $\pm x_j \geq b_i$. If $(k-b) \neq 0$, the constraints C_i, i = b+1,b+2,.....,k, are read next i.e. a_{ij} and b_i, starting first from equality constraints. The initial point $\underline{x}^0$ is chosen as described above. As in the gradient projection method, scaling of the constraints is also essential in this technique. The scaling divisor for a particular equality or inequality constraint is the square root of the

sum of the coefficients of $\underline{x}$ squared. This is essentially normalization of the constraints whereby the coefficients of $\underline{x}$ become n_{ij} and the right hand side of the constraints become b_i'.

Tolerances

The gradient tolerance, ε_1, is used to determine when the norm of the gradient is zero, and the problem has reached a maximum. The value of ε_1 is harder to determine for a nonlinear function. In general, the smaller the value of ε_1, the better the "maximum" will be. But, this will be at the cost of computation time. The value of ε_1 is chosen to be .001.

The constraint tolerance, ε_2, determines when a point is on a constraint, and is essentially an acceptable error in satisfying the constraints. ε_2 is taken = .0001.

The linear dependence tolerance, ε_3, determines when a constraint is linearly dependent and therefore cannot be added to the basis. The value of ε_3 is chosen as .005.

Linearly dependent constraints

The normal distance from the point $\underline{x}$ to a constraint C_i is called lambda $\lambda_i(\underline{x}) = \sum_{j=1}^{m} n_{ij}\, x_j - b_i'$. These lambdas are tested to determine feasibility and which constraints are satisfied as equalities. The constraint vectors form the columns of the constraint matrix $\underline{N}_k$;

$\underline{N}_k = [\underline{n}_1, \underline{n}_2, \ldots\ldots\ldots, \underline{n}_k]$.

The basis is the set of constraints which are active, or limiting, at a particular point during the solution of a problem. The initial basis consists of equalities, if any, which are, by definition, always limiting. In the course of the solution other constraints which are found to be

limiting are added to the basis, and some of these may be dropped from the basis later if they are no longer limiting. The algorithm drops or adds one constraint at a time. The number of constraints in the basis may be less than or equal to m. Those columns of the constraint matrix which are in the basis are represented by $\underline{N}_q$. Corresponding to the basis we have an inverse matrix, $(\underline{N}_q^T \underline{N}_q)^{-1}$.

The concept of linear dependence is very crucial to the overall accuracy of the solution. Since $(\underline{N}_q^T \underline{N}_q)$ must be non-singular, constraints which are linearly dependent on the constraints in the basis must not be added. Since most of the computation is based on the inverse of this matrix, it is also necessary to exclude those constraints which are almost linearly dependent in order to retain a reasonable amount of accuracy. The procedure given in Appendix A, in addition to computing the inverse matrix, results in a convenient test for linear dependence, since a constraint is linearly dependent if its projection is zero. This test is actually made against a non zero tolerance, ε_3. If $||\underline{P}\,\underline{n}_i|| \leq \varepsilon_3$, then C_i is linearly dependent and is not added to the basis.

As previously mentioned any point satisfying the equality and inequality constraints is a feasible point. When a constraint is added to the basis, it is considered as an equality when testing feasibility. Since $\underline{x}$ should remain on the constraints in the basis, this gives a good criterion for detecting round-off errors and preventing $\underline{x}$ from going astray. A linearly dependent constraint should theoretically be satisfied along with the constraints in the basis, so, to be feasible, $\underline{x}$ must also be on the linearly dependent constraint. If $\underline{x}$ is not feasible with respect to a particular constraint, then that constraint is said to be violated. The test for feasibility at any $\underline{x}$ is: If $\lambda_i(\underline{x}) \geq -\varepsilon_2$, all C_i, and

$|\lambda_i(\underline{x})| \leq \varepsilon_2$, all C_i in the basis, $\underline{x}$ is feasible. The $\underline{x}$-correction procedure given in Section 5.2 corrects $\underline{x}$ to the constraints in the basis, whereby it finds an $\underline{x}$ for which $|\lambda_i(\underline{x})| \leq \varepsilon_2$ for all C_i in the basis.

The gradient, $g(\underline{x})$, gives the direction of increasing $f(\underline{x})$. Steps are taken along $\underline{H}_q\ \underline{g}$ to increase f(x) while staying feasible. If $||\underline{Pg}|| \leq \varepsilon_1$, and $r_i/\sqrt{d_{ii}} \leq \varepsilon_1$, where i is an inequality constraint in the basis, then this is the maximum. The quantity r_i is as defined in Appendix A and d_{ii} is the ith element of the inverse matrix. It is possible to take a step which requires no changes in the basis. Such a step is called an interior step. That is, if $\underline{x}^1$ lies on the same intersection as $\underline{x}^0$, then the step to $\underline{x}^1$ was interior. When a step is interior, a test is made for a constraint to be dropped even if the gradient is not zero. This speeds up the convergence of a nonlinear problem, since a zero projected gradient does not have to be found with a basis that is not optimal.

Forming a Basis and Re-inverting

If the problem contains equalities, the initial basis consists of these equalities. The increase of this initial basis is computed when and only when new constraint data is read. When computing this initial inverse, the equalities are added in order of maximum $||\underline{P}\ \underline{n}||$. After the first equality is added, the remaining equalities are projected, and the one with the maximum $||\underline{P}\ \underline{n}||$ (most linearly independent) is added next. This process is repeated until a linearly dependent equality is found or all equalities have been added to the basis. Equalities which are linearly dependent, if any, are not added to the basis, but will still be classified as equalities rather than linearly dependent constraints.

Changes are made in the basis to find a feasible point as well as the

maximum. Briefly, the normal criteria for adding a constraint to the basis are: 1) if $\underline{x}$ is not feasible, add the constraint with the most negative lambda, and 2) if the gradient is non-zero, add the constraint with the most negative $\underline{z}^T \underline{n}_i$ where $|\lambda_i| \leq \epsilon_2$, if any. $\underline{z}$ is the unit vector in direction of step i.e. $\underline{z} = \underline{H}_q \underline{g} / ||\underline{H}_q \underline{g}||$. Constraints may be dropped from the basis for any of the following reasons: 1) $\underline{x}$ is not feasible and the constraint with the most negative lambda is linearly dependent; 2) the step was interior, and 3) the gradient is zero.

MXRN, maximum number of re-inversions. The number of re-inversions is limited to prevent the problem from re-inverting too often thereby consuming extra time. The program will not re-invert twice in a row to the same basis, but it may re-invert after only one basis change or possibly repeat a series of re-inversions.

LIMIT, maximum number of steps. The number of steps required to reach the maximum is difficult to pre-determine since it depends on such factors as size, type of function and number of constraints in the basis. The choice of maximum number of steps should be such that results are not repeated unnecessarily after the maximum is attained.

Subroutine CLASS classifies the constraints into categories v and w. In the program

u = linearly dependent constraints $||\underline{P}\, \underline{n}_i|| \leq \epsilon_3$

v = constraints not in the basis with $\lambda = 0$

w = constraint not in the basis with $\lambda > 0$

q = number of constraints in the basis

q^* = constraints added to the initial basis

Subroutine REINV essentially computes the inverse matrix, whereas subroutine COMP1 and COMP2 do the matrix computations required by the conjugate

gradient algorithm.

In the light of the above description, the computational procedure followed for the method under consideration is briefly described.

The main program is divided into three parts: input, starting procedure and the iteration procedure. The first section reads the input and obtains the initial conditions for the problem. When the constraints are read, they are normalized and stored. If equality constraints are present, then these equalities form the initial basis, and the corresponding inverse matrix is computed and stored.

The next section is the starting procedure, in which the program tests for feasibility and, if necessary, obtains a feasible $\underline{x}$. If $\underline{x}$ is not feasible, the procedure for finding a feasible $\underline{x}$ is as follows. If one or more constraints in the basis are violated, i.e. $|\lambda_i| > \varepsilon_2$, one $\underline{x}$ correction is computed. All constraints in the current constraint basis should then be satisfied. If any constraint in the basis is still violated, this is considered as an $\underline{x}$-correction failure, and this necessitates a re-inversion to proceed further. If constraints which are not in the basis are violated i.e. $\lambda_i(\underline{x}) < -\varepsilon_2$, the program computes the most negative lambda and tries to add the corresponding constraint to the basis. If this constraint is linearly independent, only then it is added, and the procedure is repeated starting from the $\underline{x}$-correction. On the other hand if the constraint is linearly dependent, the program tests against ε_3 for a constraint to be dropped. If it finds no constraint to be dropped on the initial iteration, there is no feasible $\underline{x}$. On the later iterations, this is considered to be an $\underline{x}$-correction failure. In either case, a re-inversion is essential to proceed. However, in the case of no feasible $\underline{x}$ a re-inversion will not necessarily obtain a feasible $\underline{x}$ if one does not exist. If there

is a constraint to be dropped then it is dropped from the basis. However, the program tries to add the constraint with the most negative lambda. If this constraint is still linearly dependent, it is considered to be an $\underline{x}$-correction failure and this necessitates a re-inversion to proceed. Depending upon the number of constraints in the basis the matrix $\underline{H}_q$ is computed according to the relation (4) in section 3.

On obtaining a feasible $\underline{x}$ at the beginning of each iteration, subroutine FUNCT is called upon to compute $f(\underline{x})$ and $g(\underline{x})$. This entitles the program to enter the iteration procedure. An iteration includes the projection of the gradient, testing for the maximum, changes in the basis, computation of the matrix $\underline{H}_q$ and the calculation of $\underline{z}$. The subroutine CLASS classifies the non-basis constraints into v and w. If $||\underline{g}|| \leq \varepsilon$ and $\alpha_j \leq 0$, j=e+1,e+2, ...,q, then the point is the maximum required.

If $||\underline{g}|| > \varepsilon_1$ and q=0 then $\underline{z}$ is computed. If q > 0 and $||\underline{Pg}|| \leq \varepsilon_1$ then the program makes the necessary test as given in the flow chart (Appendix C) for dropping a constraint from the basis. If there is no constraint to drop, then the maximum test is satisfied, and the current point is the optimum. On dropping the constraint, the matrix $\underline{H}_q$ is computed using the proper relation.

If $||\underline{Pg}|| > \varepsilon_1$ and (m-q) > 0, $\underline{z}$ is computed. If v > 0 and $(\min \underline{z}^T \underline{n}_i) < 0$, for i in v, then the corresponding constraint if linearly independent is added to the basis and $\underline{H}_q$ is computed. If the constraint is linearly dependent the program repeats for $(\min \underline{z}^T \underline{n}_i) < 0$, i in v. When v=0, the program tests for basis change at this iteration. If q^* or $\eta = 0$ the program tests against $2||\underline{Hg}||$ for a constraint to be dropped (step (ii) of the conjugate gradient algorithm). On dropping a constraint, the program

returns to a location where the gradient is projected and tested as given in the flow chart. If the constraint cannot be dropped, then step (iii) of the algorithm is followed, wherein the cubic interpolation scheme is followed to obtain a higher value of $\underline{x}$. If $\gamma^i = \lambda^i$ as in step (iv), then the corresponding constraint of step (iii) is added to the basis and $\underline{H}_q$ is computed. If step (iv) fails, proceed to step (iv). If step (iv) fails, then step (v) is executed. At the beginning of step (iii), the iteration counter is changed. The program stops if the number of iterations exceeds the limit given.

5. APPLICATION OF THE CONJUGATE GRADIENT METHOD TO MANAGEMENT PROBLEMS

Faced with the increasing complexities of engineering, production and sales in a highly developed and often strongly competitive economy, management must deal with a multitude of interlocking and often far-reaching tasks involving many decision-making and control problems. In many industries the control of inventory is intimately related to the scheduling of production, and in this section of the report two different production scheduling and inventory models are considered.

The first one is a simple production planning model involving a single state variable. The second is a more complex model consisting of two state variables. Solutions were obtained by the application of the conjugate gradient technique.

5.1 Test Problem No. 1

An inventory model with one state variable

Consider the case of a manufacturing enterprise whose sales rate is known with certainty. The rate of change of inventory level I(t) is given by

$$\frac{dI(t)}{dt} = P(t) - Q(t) \tag{1}$$

where $P(t)$ = production rate at time t

and $Q(t)$ = sales rate at time t.

The problem is to minimize the cost function given by

$$C_T = \int_0^T [C_I (I_m - I(t))^2 + C_P \exp (P_m - P(t))^2] \, dt \tag{2}$$

where C_T is the total cost of production and inventory. C_I is the cost of carrying inventory and C_P is the minimum production cost which occurs when the production rate equals P_m. The quantity P_m can be considered as the production capacity of the manufacturing plant. Since the plant is designed for a capacity P_m, an increase in production may require additional equipment and manual labor which can be very expensive. On the other hand, a decrease in production below P_m will be equally expensive due to maintenance of unused equipment and idle labor which cannot be decreased due to contract agreements. The quantity I_m can be considered as the capacity for storage of inventory. In actual practice, the minimum storage cost is obtained when the storage capacity is completely utilized. In addition, the cost function given by equation (2) has the smoothing capability which is frequently desirable for many manufacturing processes. In this case, I_m and P_m can be considered as desirable inventory and production levels. Let us further assume that the sales forecast is known and is given by the linear relation

$$Q(t) = a + bt \tag{3}$$

and the initial inventory is

$$I(0) = c \tag{4}$$

where a, b and c are known constants.

Substituting for Q(t) in equation (1) we have

$$\frac{dI(t)}{dt} = P(t) - a - bt \tag{5}$$

In order to solve this model by the conjugate gradient algorithm, equations (2) and (5) are approximated by difference equations.

Thus, equation (5) is reduced to