REVUE FRANÇAISE D'AUTOMATIQUE, INFORMATIQUE, RECHERCHE OPÉRATIONNELLE. RECHERCHE OPÉRATIONNELLE

LEON S. LASDON
RICHARD L. FOX
MARGERY W. RATNER

Nonlinear optimization using the generalized reduced gradient method

Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle, tome 8, n° V3 (1974), p. 73-103

<http://www.numdam.org/item?id=RO_1974__8_3_73_0>

R.A.I.R.O.
(8e année, novembre 1974, V-3, p. 73 à 104)

NONLINEAR OPTIMIZATION USING THE GENERALIZED REDUCED GRADIENT METHOD (1)

by Leon S. LASDON, Richard L. FOX and Margery W. RATNER

Abstract. — *This paper describes the principles and logic of a system of computer programs for solving nonlinear optimization problems using a Generalized Reduced Gradient Algorithm. The work is based on earlier work of Abadie (2). Since this paper was written, many changes have been made in the logic, and significant computational experience has been obtained. We hope to report on this in a later paper.*

1. INTRODUCTION

Generalized Reduced Gradient methods are algorithms for solving nonlinear programs of general structure. This paper discusses the basic principles of GRG, and constructs a specific GRG algorithm. The logic of a computer program implementing this algorithm is presented by means of flow charts and discussion. A numerical example is given to illustrate the functioning of this program.

2. BASIC IDEAS OF GRG

The nonlinear program to be solved is assumed to have the form

$$\text{minimize } f(X) \tag{1}$$

$$\text{subject to } g_i(X) = 0 \quad , \quad i = 1, ..., m \tag{2}$$

$$l_i \leqslant X_i \leqslant u_i \quad , \quad i = 1, ..., n \tag{3}$$

(1) This report was prepared as part of the activities of the Department of Operations Research, School of Management and the Solid Mechanics, Structures, and Mechanical Design Division, School of Engineering, Case Western Reserve University (under Contract N0014-67-A-0404-0010 with the Office of Naval Research). Reproduction in whole or part is permitted for any purpose by the United States Government.
Technical Memorandum No. 325.

(2) The authors are indebted to Jean Abadie for his many helpful suggestions.

where X is n-vector and u_i, l_i are given lower and upper bounds $u_i > l_i$. We assume $m < n$ since, in most cases, $m \geqslant n$ implies an infeasible problem or one with a unique solution. The form (1)-(3) is completely general, since inequality constraints may always be transformed to equalities, as in (2), by the addition of slack variables. The vector X contains as components both the « natural » variables of the problem and these slacks.

The fundamental idea of GRG is to use the equalities (2) to express m of the variables, called *basic variables*, in terms of the remaining *n-m nonbasic* variables. This is also the way the Simplex Method of linear programming operates. Let $\bar{X}$ be a feasible point and let y be the vector of basic variables and x the nonbasic at $\bar{X}$, so that X is partitioned as

$$X = (y, x), \bar{X} = (\bar{y}, \bar{x}) \tag{4}$$

and the equalities (3) can be written

$$g(y, x) = 0 \tag{5}$$

where

$$g = (g_1, ..., g_m) \tag{6}$$

Assume that the objective f and constraint functions g_i are differentiable. Then, by the implicit function theorem, in order that the equations (5) have a solution $y(x)$ for all x in some neighborhood of $\bar{x}$, it is sufficient that the $m \times m$ *basis matrix* $\partial g/\partial y$, evaluated at $\bar{X}$, be nonsingular.

Assume that it is. Then the objective may be expressed as a function of x only :

$$F(x) = f(y(x), x) \tag{7}$$

and the nonlinear program is transformed, at least for x close to $\bar{x}$, to a *reduced problem* with only upper and lower bounds :

$$\text{minimize } F(x) \tag{8}$$

subject to

$$l_{NB} \leqslant x \leqslant u_{NB} \tag{9}$$

where l_{NB} and u_{NB} are the vectors of bounds for x. GRG solves the original problem (1)-(3) by solving a sequence of problems of the form (8)-(9). Such problems may be solved by simple modifications of unconstrained minimization algorithms.

For the reduced problem (8)-(9) to yield useful results, it is necessary that x be free to vary about the current point $\bar{x}$. Of course, the bounds (9) restrict x, but it is easy to move x in directions which keep these bounds satisfied. The bounds on the basic variables, however, pose a more serious problem. If some components of $\bar{y}$ are at their bounds, then even a slight change in x from $\bar{x}$ may

cause some bound to be violated. To guarantee that this cannot happen, and to insure the existence of the function $y(x)$, we assume that the following condition holds :

Nondegeneracy Assumption

At any point X satisfying (2)-(3), there exists a partition of X into m basic variables y and n-m nonbasic variables x such that

$$l_B < y < u_B \tag{10}$$

where l_B and u_B are the vector of bounds on y and

$$B = \partial g/\partial y \text{ is nonsingular} \tag{11}$$

This assumption is quite mild, as we show later.

Consider now starting from the feasible point $\bar{X}$ with basic variables y and nonbasic variables x, and attempting to solve the reduced problem (8)-(9). By (7), to evaluate the objective $F(x)$, we must know the values of the basic variables $y(x)$. Of course, except for linear and a few nonlinear cases, the function $y(x)$ cannot be determined in closed form. However, $y(x)$ can be computed for any given x by an iterative method which solves the equalities (5). Hence a procedure for solving the reduced problem starting from $X_0 \equiv \bar{X}$, is

(0) set $i = 0$.

(1) Substitute x_i into (5) and determine the corresponding values for y, y_i, by an iterative method for solving nonlinear equations.

(2) Determine a direction of motion, d_i, for the nonbasic variables x.

(3) Choose a step size α_i such that

$$x_{i+1} = x_i + \alpha_i d_i$$

This is often done by solving the one dimensional search problem

$$\text{minimize } F(x_i + \alpha d_i)$$

with α restricted such that $x_i + \alpha d_i$ satisfies the bounds on x. This one dimensional search will require repeated applications of step (1) to evaluate F for various α values.

(4) Test the current point $X_i = (y_i, x_i)$ for optimality. If not optimal, set $i = i + 1$ and return to (1).

If, in step (1), the value of one or more components of y_i exceed their bounds, the iterative procedure must be interrupted. For simplicity, assume only one basic variable violates a bound. Then this variable must be made nonbasic and some component of x which is not on a bound is made basic. After this *change of basis*, we have a new function $y(x)$, a new function $F(x)$,

and a new reduced problem. These ideas are illustrated geometrically in Figure 2.1. The initial point $\bar{X}$ is on the curve $g_2(X) = 0$. We have taken the

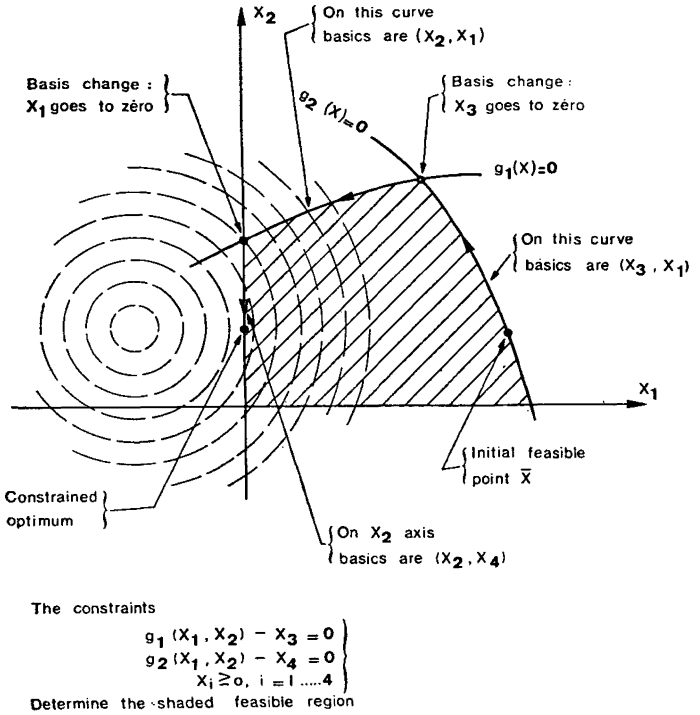


Figure 2.1
Sequence of Basis Changes

basic variables as (x_3, x_1), although the only variable that cannot be basic is x_4, since it is at lower bound of zero. The objective of the first reduced problem is $F_1(x_2, x_4)$, which is just the objective f as measured on the curve $g_2 = 0$. It is possible that the algorithm minimizing F_1 might release x_4 from its lower bound of zero, in which case we would move interior to $g_2 = 0$. Assume, for purposes of illustration, that this does not happen. Then we move along $g_2 = 0$ as indicated by the arrow until we hit the curve $g_1 = 0$. At this point the slack for g_1, x_3, goes to zero. Since it is basic, it must leave the basis, to be replaced by one of the nonbasics, x_2 or x_4. Since x_4 is zero, x_2 becomes basic. Now we have a new objective, $F_2(x_3, x_4)$, with x_3 and x_4 at lower bound of zero. The algorithm optimizing F_2 will determine that, if either x_3 or x_4 is released from its lower bound, F_2 can be decreased. Assume x_4 is released from its bound (actually x_3 and x_4 might both be released from their bounds). Then the algorithm will begin to minimize F_2, which is simply f as measured along the curve $g_1 = 0$. Motion is towards the x_2 axis. Upon reaching it,

x_1 becomes zero, and another basis change occurs, with x_1 becoming nonbasic and x_4 becoming basic. Then optimization of the new function $F_3(x_1, x_3)$ will terminate at the constrained optimum of f.

The Reduced Gradient

GRG can be implemented without using derivatives of f or the g_i. This requires methods for solving nonlinear equations and for minimizing nonlinear functions subject to bounds which do not use derivatives. Although such methods exist, there are a much greater variety which do require derivatives. The efficiency of these is better understood, and their use in large problems is better established.

Hence we concern ourselves from now on with GRG algorithms which require first derivatives of f and g.

In minimizing F using derivatives, we must have a formula for ∇F. F is guaranteed to be differentiable if f and g are, and if $\partial g/\partial y$ is nonsingular, since then the implicit function $y(x)$ is differentiable. By (7)

$$\partial F/\partial x_i = \partial f/\partial x_i + (\partial f/\partial y)^T \, \partial y/\partial x_i \tag{12}$$

To evaluate $\partial y/\partial x_i$, use the fact that, if

$$g_j(y(x), x) = 0 \quad , \quad j = 1, ..., m$$

for all x in some neighborhood of $\bar{x}$, then

$$\mathrm{d}g_j/\mathrm{d}x_i = 0 = (\partial g_j/\partial y)^T \, \partial y/\partial x_i + \partial g_j/\partial x_i \quad , \quad j = 1, ..., m$$

or, in matrix form

$$(\partial g/\partial y) \, \partial y/\partial x_i + \partial g/\partial x_i = 0$$

Since $(\partial g/\partial y)$ is nonsingular at $\bar{X}$

$$\partial y/\partial x_i = -(\partial g/\partial y)^{-1} \, \partial g/\partial x_i \equiv B^{-1} \, \partial g/\partial x_i \tag{13}$$

Using (13) in (12)

$$\partial F/\partial x_i = \partial f/\partial x_i - (\partial f/\partial y)^T B^{-1} \, \partial g_k \partial x_i \tag{14}$$

Let

$$\pi = (\partial f/\partial y)^T B^{-1} \tag{15}$$

As is shown later, the m-vector π is the Kuhn-Tucker multiplier vector for the constraints g. Using (15), the components of ∇F are

$$\partial F/\partial x_i = \partial f/\partial x_i - \pi^T \, \partial g/\partial x_i \tag{16}$$

Equation (16) reduces to the formula for the relative cost factors in linear programming [1] if f and all g_i are linear. Then, $\partial f/\partial x_i = c_i$, $\partial f/\partial y = c_B$ (the

objective coefficients of the basic variables) and $\partial g/\partial x_i = P_i$, the column of constraint coefficients for x_i. The vector π is the simplex multiplier vector.

Relation of Reduced Gradient Formula and Kuhn-Tucker Conditions

If $\bar{X}$ is optimal for (1)-(3), and if the gradients of all binding constraints of $\bar{X}$ are independent (see [2]), then the Kuhn-Tucker conditions hold at $\bar{X}$. To write these, let π be a Lagrange multiplier vector for the equalities (2), and α and β be multipliers for the lower and upper bound constraints respectively. The Lagrangian for (1)-(3) is

$$L = f + \pi g + \alpha(l - X) + \beta(X - \pi)$$

The Kuhn-Tucker conditions, written in terms of y and x, are

$$\partial L/\partial y = \partial f/\partial y + \pi B - \alpha_y + \beta_y = 0 \tag{17}$$

$$\partial L/\partial x = \partial f/\partial x + \pi \partial g/\partial x - \alpha_x + \beta_x = 0 \tag{18}$$

$$\alpha \geqslant 0, \quad \beta \geqslant 0 \tag{19}$$

$$\alpha(l - x) = \beta(x - \pi) = 0 \tag{20}$$

where α_y, β_y are subvectors of α and β corresponding to the basic variables y, and similarly for α_x, β_x. If $\bar{X}$ is optimal, there exist vectors $\bar{\pi}$. $\bar{\alpha}$, $\bar{\beta}$ which, together with $\bar{X}$, satisfy (17)-(20). Since $\bar{y}$ is strictly between its bounds, (20) implies

$$\alpha_y = \beta_y = 0$$

Then (17) implies

$$\pi = -\partial f/\partial y B^{-1}$$

so the vector π in (15) is the multiplier vector for the equalities (2).

Then (18) may be written

$$\partial f/\partial x + \pi\, \partial g/\partial x = \alpha_x - \beta_x \tag{21}$$

The left hand side of (21) is simply the reduced gradient, $\nabla F(x)$. To relate (21) to the problem (8)-(9), if x_i is strictly between its bounds then $\alpha_{x_i} = \beta_{x_i} = 0$ by (20), so

$$\partial F/\partial x_i = 0 \tag{22}$$

If x_i at lower bound then $\beta_{x_i} = 0$ so

$$\partial F/\partial x_i = \alpha_{x_i} \geqslant 0 \tag{23}$$

while if x_i is at upper bound, $\alpha_{x_i} = 0$ so

$$\partial F/\partial x_i = -\beta_{x_i} \leqslant 0 \tag{24}$$

But (22)-(24) are just the optimality conditions for the reduced problem (8)-(9). Hence the Kuhn-Tucker conditions for (1)-(3) may be viewed as optimality conditions for the reduced problem (8)-(9), and π in the formula for the reduced gradient is the Kuhn-Tucker multiplier vector. This vector is useful for sensitivity analysis, and GRG provides it as a by-product of its computations.

Relation of Nondegeneracy Assumption and Luenberger Constraint Qualification

Let X^0 be an optimal solution to (1)-(3). Luenberger [2] has shown that a sufficient condition for the Kuhn-Tucker conditions to hold at X^0 is that the gradients of all binding constraints be linearly independent.

Assume that this is the case. Then, at most n of the $2n + m$ constraints (1)-(3) can be binding at X^0. Since all m of the equalities (2) are binding, at most n-m of the constraints (3) can be binding, i.e at most n-m of the variables X_i can be at a bound. Hence there will be at least m variables X_i satisfying $l_i < X_i^0 < u_i$. Consider now the Jacobian matrix of the binding constraints evaluated by X^0. If all variables not at bounds are grouped together, this Jacobian has the structure

	$k > m$ variables not on bounds	$n - k$ variables on bounds
m equality rows	j_1^0	j_2^0
$n - k$ bounds rows	0	I_{n-k}

Since this matrix must have all $m + (n - k)$ rows linearly independent, it must have this same number of independent columns. Since the $n - k$ rightmost columns are independent, the submatrix j_1^0 must contain m independent columns. Let B^0 be a nonsingular $m \times m$ submatrix chosen from j_1^0, and let y be the m-vector of variables associated with the columns of B^0, with x the vector of the remaining $n - m$ variables. Then $l_B < y^0 < u_B$ and B^0 is nonsingular.

That is, the nondegeneracy assumption stated earlier is true at X^0, so it is implied by Luenbergers constraint qualification. This information is useful, since Luenbergers qualification appears to be satisfied at the optimum, indeed at all feasible points, of all but a few pathological nonlinear programs. However, problems can arise where the binding constraint gradients become nearly dependent, and then B becomes nearly singular, and its inversion and other operations with it become numerically unstable. A computer program implementing GRG must test for this near-singularity and attempt to correct it if it occurs.

3. A GRG ALGORITHM

In this section we decribe the GRG algorithm developed during the period Nov. 1972-Nov. 1973. The major differences between this algorithm and the procedure described by Abadie in [3] and [4] are :

1. The algorithm works only with the currently active constraints. Since, in most problems, not all constraints are active, this can ease computations considerably. The basis matrix has a row for each active constraint, and changes size as constraints are encountered or dropped. Gradients of inactive constraints are not required, a significant advantage in problems with many constraints.

2. The algorithm used to optimize the objective on each constraint intersection is the Davidon-Fletcher-Powell (DFP) method [3], modified to account for upper and lower bounds. This should yield more rapid convergence than the gradient or conjugate gradient procedures used by Abadie.

3. A new procedure has been constructed for deciding whether to incorporate a constraint into the current constraint basis. The constraint is incorporated if the one-dimensional minimum currently being sought is on the boundary of the current constraint intersection. Mechanisms for determining this efficiently in the context of GRG have been developed.

4. A new basis change procedure is used. In [2], Abadie makes a basis change if a basic variable violates one of its bounds during the Newton iteration. This can lead to « false » basis changes if the Newton algorithm is not converging, or is converging but not monotonically. We wait until the Newton algorithm has converged, then treat the violated bound as a newly encountered constraint, and apply the procedures in (3) above. This insures that the objective value after a basis change is lower than all previous values (this is not true in Abadies realization).

5. The one-dimensional search is the core of the algorithm and is crucial to its efficiency. We have adopted the algorithm described in [6] to operate within GRG. This procedure is the result of many years of development, and computational results using it in unconstrained minimization have been excellent.

We new present and discuss flow charts of our GRG algorithm and, in this context, discuss the above ideas in more detail. The algorithm currently requires a feasible starting point. Work during the next year will include designing a phase I procedure, which will find a feasible point or determine that none exists.

Let $\bar{X} = (\bar{y}, \bar{x})$ be a feasible point for the constraints (2)-(3). Further, suppose that the first m_1 constraints are equalities, and the remaining m_2 are inequalities, with $m_1 + m_2 = m$. That is, the constraints may be written :

$$g_i(X) = 0, \quad i = 1, ..., m_1$$
$$g_i(X) \geqslant 0, \quad i = m_1 + 1, ..., m$$

We define the index set of binding constraints at $\bar{X}$ as

$$\text{IBC} = \{ i \mid g_i(\bar{X}) = 0 \}$$

The surface defined by a set of active constraints will be called the constraint intersection, S :

$$S = \{ X \mid g_i(X) = 0, \quad i \in \text{IBC} \}$$

GRG moves from one such surface to another as new constraints become binding and previously binding constraints become positive. In our realization of GRG, while on a particular constraint intersection, we ignore the positive constraints, except for evaluating them at each point to check that they are still positive. The constraint basis at $\bar{X}$ contains a row for each index in IBC. Since the slacks of binding constraints are zero (i.e. at lower bounds), there will never be any slacks in the basis. If there are NB binding constraints, the NB basic variables are all chosen from the n « natural » variables, $X_1, \ldots, X_n$, while the n nonbasics are the remaining n — NB natural variables, plus the NB slacks of the binding constraints.

Use of Goldfarb Variable Metric Algorithm

Since GRG requires at least partial solution of a number of reduced problems of the form (8)-(9), each algorithm for solving such problems leads to a variant of GRG. The choice of algorithm is critical, since once GRG approaches an optimum, and no more basis changes occur, its convergence rate is that of the algorithm selected. It would be foolish to use the method of steepest descent, as its convergence rate is at best geometric, with a very small convergence ratio for problems whose Hessian at the optimum is badly conditioned [2]. The conjugate gradient method used by Abadie is superlinearly convergent [2], but many computational experiments have shown it to be considerably slower in practice (in terms of number of iterations) than methods of the variable metric class [2]. For this reason, we have chosen the variable metric method of Goldfarb [5], simplified for the special case of bounded variables, to solve the reduced problems.

The flow chart entitled « Main GRG Program » illustrates our adaptation of Goldfarbs algorithm. The algorithm is very much as described in [5], but simplified for the special case of bounded variables (see [7]). The flow chart is almost exactly as it would be if the only constraints present were the bounds on the nonbasic variables. All the logic required to deal with the nonlinear constraints (2) is in the one dimensional search subroutine on page 2 of the chart. The algorithm chooses search directions by the formula

$$d_i = - H_i \nabla F(x_i)$$

where H_i is an $n \times n$ symmetric positive semi-definite matrix. This matrix projects any vector onto the bounds, i.e, for any vector v, Hv is zero in the i^{th}

position if x_i is at a bound. The initialization in block 1, page 1, and the updating of block 3, page 2 (which forces row and column r of H to zero when x_r hits a bound) guarantee that H always has this property. The algorithm will minimize a positive definite quadratic objective, subject to upper and lower bounds, in a finite number of iterations. If at the optimum, $n_1 \leqslant n$ of the variables are at their bounds, then, once these variables reach their bounds, the optimal values of the remaining ones will be found in at most $n - n_1$ iterations. Further, the nonzero rows and columns of H form a positive definite matrix which will converge (in the case of quadratic objective) to the inverse Hessian of the function of $n - n_1$ variables formed by replacing the n_1 variables at bounds by the values of those bounds.

Block 2, page 1 of the flow chart, is performed as follows : The Kuhn-Tucker multiplier for the lower bound constraint on x_i is $\lambda_i^l = \partial F/\partial x_i$, and for the upper bound $\lambda_i^u = -\partial F/\partial x_i$. If x_i is at lower (upper) bound and $\lambda_i^l(\lambda_i^u)$ is negative, then F can be decreased by increasing (decreasing) x_i, i.e. by moving away from the bound. In our program, from all variables at bounds whose multipliers are negative, we choose the variable with the multiplier of largest absolute value. If this value is larger than twice $\|d_i\|$ (where $\|\ \|$ indicates Euclidean norm), we allow this variable to leave its bound by setting the corresponding diagonal element of H (currently zero) to one. This causes $\|d_i\|$ to increase. We then test the remaining bounded variables for negative multipliers, and repeat the above procedure. The test against $\|d_i\|$ insures that we do not leave a constraint subspace until $\|d_i\|$ becomes « small enough ». This helps to prevent zigzagging where we constantly leave and then return to the same subspace.

Goldfarbs algorithm provides search directions for the one dimensional search subroutine, in which the variables of the problem are assigned new values. This subroutine finds a first local minimum for the problem

$$\text{minimize } F(\bar{x} + \alpha d)$$

where F is the reduced objective, $\bar{x}$ the initial values of the nonbasic variables, and d the search direction. The iteration subscript, i, has been dropped for convenience. The direction d is always a direction of descent, i.e.

$$d^T \nabla F(\bar{x}) < 0$$

The procedure starts with a search for three α values, A, B, and C, which satisfy

$$0 \leqslant A < B < C$$

$$F(\bar{x} + Ad) > F(\bar{x} + Bd)$$

and

$$F(\bar{x} + Cd) > F(\bar{x} + Bd)$$

Then the interval $[A, C]$ contains a local minimum of $F(\bar{x} + \alpha d)$. In block 18 of page 1 of the one dimensional search flow chart, a quadratic is passed through A, B and C, with its minimum at D. On page 3 of the flow chart, the points A, B, C, D are used to initiate an iterative cubic interpolation process which yields the final α value.

The logic on the first two pages of the flow chart locates the points A, B, C. In doing this, the choice of initial step size, α_0, (block 1, page 1), is important. With Goldfarbs algorithm or other variable metric methods, α_0 is set equal to the optimal α value from the previous search except when this causes too large a change in the variables. The theoretical basis for this is that, as a variable metric converges, the optimal α values should converge to 1, the optimal step for Newton's Method. Hence the previous optimal step is a good approximation to the current one. This must be modified when the method is restarted, for example when a new constraint is encountered or the basis is changed, since then an optimal step much less than unity is generally taken. Hence, we require that the change in any nonbasic variable larger than 10^{-3} in absolute value not exceed .05 times its value, while the change in any variable smaller than 10^{-3} in absolute value cannot exceed 0.1. If the largest α value meeting these conditions is α^1, then, at iteration i, α_0 is given by

$$\alpha_0 = \min(\alpha_{i-1}, \alpha^1)$$

The loop 2, 3, 4, 5 halves the step size until a value $FB < FA$ is achieved, or until SLPFLG = 0. The variable SLPFLG is initialized at zero in subroutine NEWG and is set to unity in block 3 of the NEWG flow chart if a new constraint has been encountered and the minimum along d is interior to that constraint. The test in block 6 of the one dimensional search flow chart is false only if the step size has been halved at least once in 2, 3, 4, 5, in which case K1 is the function value corresponding to C. The test in block 7 prevents the subroutine from trying to pass a quadratic through 3 points which are too widely separated in function value. It also insures that the subroutine will cut back the step size if a large function value is returned by block 3. This is used to force a cutback when the NEWTON algorithm in block 3 does not converge, by setting FB to 10^{30}. The test on FC in block 12 has the same purpose. If K1 is too large, then block 8 generates a new C point 1/3 of the distance from B to C. Then the loop 8, 9, 10, 11, 12 is traversed until FC is not too large.

With $R = 0$ (which occurs if and only if (*a*) a $K1$ or FC which was too large has never been generated, or (*b*) SLPFLG = 0, the loop 10-14 transforms the points, A, B, C in figure 3.1 (*a*) into those shown in figure 3.1 (*b*). The step size is doubled each time until the points A, B, C bracket the minimum. If $K1$ or FC ever becomes too large, or if SLPFLG = 1, R is set to 1 (block 9). Then (10)-(14) transforms points as shown in figures 3.2 (*a*) thru 3.2 (*c*). Instead of doubling the step, a constant increment, $B - A$, is added. Since FC may have been

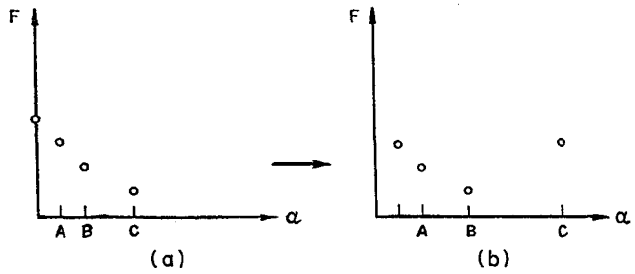


Figure 3.1

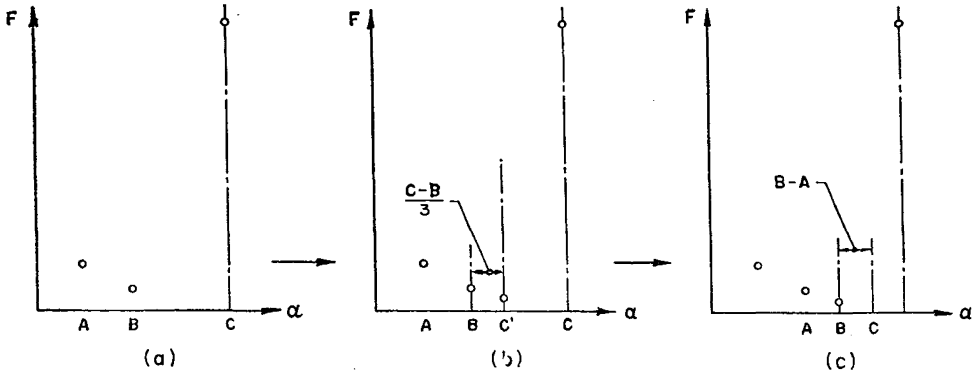
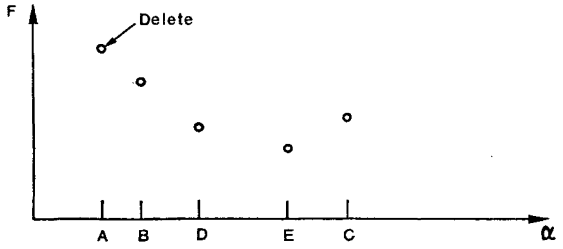
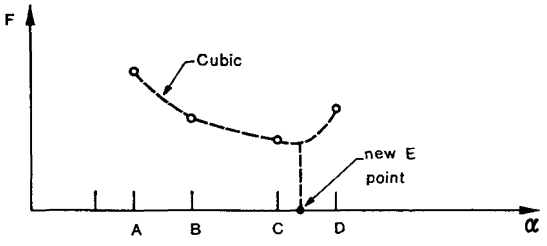


Figure 3.2



(a) -End point deleted



(b) -New 4 point pattern

Figure 3.3

Eliminating End Point in Cubic Fit

set to 10^{30} when NEWTON did not converge in block 11, there must be a provision for resetting R to 0 when 3 steps of size $(C\text{-}B)/3$ have been taken. This is accomplished by blocks 15, 16, and 17.

The quadratic interpolation in block 18 yields a fourth point, D, with function value FM, somewhere between A and C. In block 19, a cubic polynomial is passed through the 4 points FA, FB, FC, FM, and its minimum is located at the point E. The optimality tests in block 20 are passed if the percent difference between (*a*) the F values at the current and previous interpolated points and (*b*) the values of F and the cubic at E are sufficiently small. Currently $\epsilon = 10^{-4}$. In block 21 the usual situation is as shown in figures 3.3 (*a*) and 3.3 (*b*). Removal of an end point leaves 4 points which bracket the minimum and these are used in the next cubic fit. If a bracket cannot be formed by removal of an end point, the highest end point is discarded and a new cubic fit is performed. Such a situation is shown in figure 3.4.

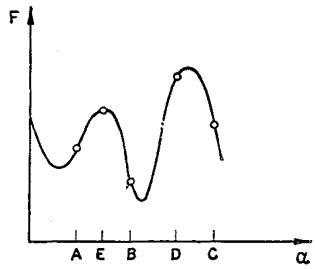


Figure 3.4

If the optimality test in block 20 is passed, and the point E is less than $\bar{\alpha}$, then E is accepted as optimal. Otherwise, F is evaluated at $\bar{\alpha}$ in block 22. If its value there is smaller than $F(\bar{x})$, $\bar{\alpha}$ is returned. If not, and $FC > FB$, the quadratic interpolation block is entered; otherwise, the subroutine terminates with an error message.

The blocks in the one dimensional search flow chart labeled « Evaluate $F(\bar{x} + \alpha d)$ » are where the basic variables are changed in response to changes in the nonbasics. As shown in the accompanying flow chart, this block contains 2 subroutines. Subroutine REDOBJ uses a Newton iteration to find new values for the basic variables y, given the values $\bar{x} + \alpha d$ for the nonbasics. It also checks for violations of currently nonbinding constraints and of bounds on basic variables, using subroutine BSCHNG to deal with these bounds.

Subroutine NEWG decides whether or not to stay on a newly encountered constraint. We now proceed to discuss these subroutines.

Subroutine REDOBJ (REDuced OBJective) is the first routine discussed

thus far that is not very similar to a procedure for unconstrained minimization. Its input is a vector of nonbasic variables $x = \bar{x} + \alpha d$, where α is the current trial value of the step size, and $\bar{x}$ is the vector of nonbasic variables at the start of the one dimensional search. It solves the system of NB nonlinear equations

$$g_i(y, \bar{x} + \alpha d) = 0 \quad , \quad i \in \text{IBC}$$

for the NB basic variables y. As in [3] and [4], this is done using the pseudo-Newton algorithm

$$y_{t+1} = y_t - B^{-1}(\hat{X}) g_B(y_t, \bar{x} + \alpha d) \quad , \quad t = 0, 1, \ldots$$

where g_B is the vector of binding constraints.

The algorithm is called pseudo-Newton because B^{-1} is not re-evaluated at each step of the algorithm, as in the standard Newton method. When α is given its first trial value in a particular one dimensional search, $\hat{X}$ is equal to the feasible point $\bar{X}$ with which we began the search. As long as Newton converges, $\hat{X}$ remains equal to $\bar{X}$, at least until the search is over. If Newton fails to converge, $\hat{X}$ may be set equal to the most recent feasible point, and B^{-1} is recomputed.

To explain blocks 1 and 2 on page 1 of the REDOBJ flow chart, consider the tangent plane to the constraint surface at $\bar{X}$. This is the set of all vectors (a, b) satisfying

$$(\partial g/\partial y)a + (\partial g/\partial x)b = 0$$

where all partial derivative matrices are evaluated at $\bar{X}$. In GRG, the change in x, b, is given by

$$b = \alpha d$$

The corresponding vector a is called the tangent vector, v. Since any scale factor multiplying v is unimportant, we may as well take $\alpha = 1$, yielding

$$v = (\partial g/\partial y)^{-1}(\partial g/\partial x)d \tag{25}$$

In our program, v is computed at $\bar{X}$, the initial point of the one dimensional search. This vector is used to find initial values, y_0, by the formula

$$y_0 = \bar{y} + \alpha_1 v$$

as illustrated in figure 3.5. Using these initial values, Newton finds the feasible point X_1. Then, at X_1, v is not recomputed. The old v is used, but emanating now from X_1, to yield the next set of initial values as

$$y_0 = y_1 + (\alpha_2 - \alpha_1)v$$

Using these, Newton finds the point X_2 of figure 3.5. This procedure is repeated until Newton fails to converge (or until the one dimensional search is over), whereupon v is recomputed at the last feasible point.

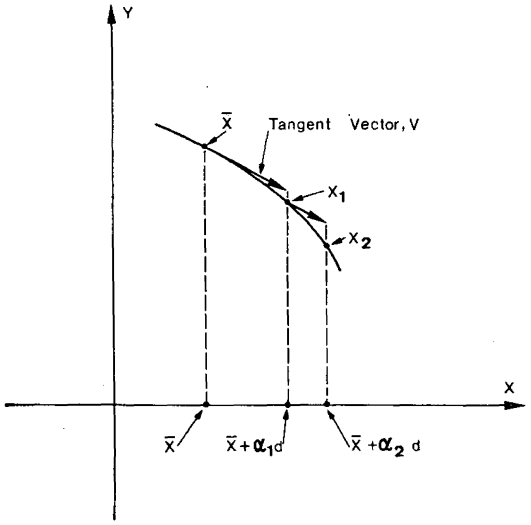


Figure 3.5

Tangent Vector Estimate of Basic Variables

Both this logic and the logic of computing B^{-1} have the objective of computing derivatives of f and the binding g_i, and of inverting B, only when absolutely necessary. If Newton converges at each point of a one dimensional search, then no derivatives or matrix inversions are required during the search.

Newton is considered to have converged if the condition

$$\text{NORMG} = \max_{i \in IBC} |g_i(X_t)| < \text{EPSNEWT}$$

is met within ITLIM iterations. Currently (using single precision arithmetic; 8 decimal digits), EPSNEWT = 10^{-4} and ITLIM = 10. If NORMG has not decreased in any set of 5 consecutive iterations (or the above condition is not met in 10 iterations) Newton has not converged, and the 2 alternatives on page 1 are tried, in an effort to achieve convergence.

The first alternative is tried if the gradients of the objective and the binding constraints have not yet been computed at the last feasible point, XPREV. These gradients are evaluated, and an approximation to the true B^{-1} at XPREV is computed, using the current B^{-1} (evaluated at some earlier feasible point) and the partial derivatives evaluated at XPREV. This approximate inverse is computed as follows :

Let B be the basis matrix evaluated at XPREV, and B_0^{-1} the basis inverse evaluated at some other point. The approximation to B^{-1} is based on the identity

$$B = B_0 + (B - B_0)$$
$$= B_0(I - B_0^{-1}(B_0 - B)) \tag{26}$$

Let

$$A = B_0^{-1}(B_0 - B) \tag{27}$$

Then, taking the inverse of both sides of (25) yields

$$B^{-1} = (I - A)^{-1}B_0^{-1} \tag{28}$$

If the points at which B_0 and B are evaluated are sufficiently close together, the norm of A will be less than unity, and $(I - A)^{-1}$ can be expanded in a power series :

$$(I - A)^{-1} = I + A + A^2 + \ldots \tag{29}$$

This series can be used to approximate the Newton correction

$$\delta = B^{-1}G \tag{30}$$

where G is the vector of binding constraints. Using (28) and (29) in (30) yields

$$\delta = [I + A + A^2 + \ldots]B^{-1}G_0$$

The i^{th} order approximation to δ, δ_i, is obtained by truncating the series expansion above at the term A^i :

$$\delta_i = \sum_{j=0}^{i} A^j B_0^{-1}G \quad , \quad i = 0, 1, 2, \ldots$$

The vectors δ_i may be determined recursively as follows :

$$\delta_0 = B_0^{-1}G$$

$$\delta_1 = (I + A)\delta_0 = \delta_0 + A\delta_0$$

$$\delta_2 = (I + A + A^2)\delta_0 = \delta_0 + A(I + A)\delta_0 = \delta_0 + A\delta_1$$

In general

$$\delta_{j+1} = \delta_0 + A\delta_j \quad , \quad j = 0, 1, \ldots, i$$

or using the definition of A in (27)

$$\delta_{j+1} = \delta_0 + \delta_j - B_0^{-1}B\delta_j \quad , \quad j = 0, 1, \ldots, i \tag{31}$$

Returning to alternative 1 on page 2 of the flow chart, this alternative is implemented by choosing the order of approximation $i (i \geqslant 1)$ and, within the

Newton subroutine, approximating δ as δ_i using the recursion (31). For $i = 1$, this is the approximation suggested by Abadie in [3]-[4].

If, after trying alternative 1, Newton again fails to converge, alternative 2 is tried. This alternative computes the true B^{-1} at XPREV, uses it in (25) to compute a new tangent vector, and returns to the Newton subroutine. If Newton still fails to converge, the final alternative is tried : the objective is set to a very large value (10^{30}), and we leave subroutine REDOBJ, returning to the one dimensional search subroutine. The large objective value will cause the search subroutine to decrease α. This will occur either in block 3 of block 11 of the subroutine. This cutback procedure will continue until Newton converges. This must happen eventually, since Newton will converge if the initial point is close enough to the solution.

Once Newton has converged, we check the positive g_i constraints to see if any are now binding or violated. Let the point Newton has obtained be

$$\hat{X} = (y(\hat{\alpha}) \quad , \quad \bar{x} + \hat{\alpha} d)$$

where

$$g_i(y(\hat{\alpha}) \quad , \quad \bar{x} + \hat{\alpha} d) = 0 \quad , \quad i \in \text{IBC}$$

Assume that some nonbinding constraints are violated;

$$g_i(\hat{X}) < 0 \quad , \quad i \in \text{IVC}$$

The program attempts to find the point at which the first nonbinding constraint went to zero. That is, we wish to find the smallest value of α, α^*, such that all constraints in *IBC* are binding, exactly one constraint from IVC is binding, and all other constraints in IVC are positive. Hence α^* satisfies

$$\begin{aligned} g_i(y(\alpha) \quad , \quad \bar{x} + \alpha d) &= 0 \quad , \quad i \in \text{IBC} \\ g_k(y(\alpha) \quad , \quad \bar{x} + \alpha d) &= 0 \end{aligned} \tag{32}$$

and

$$g_i(y(\alpha) \quad , \quad \bar{x} + \alpha d) > 0 \quad , \quad i \in \text{IVC} \quad , \quad i \neq k$$

where $k \in$ IVC. This is illustrated in figure 3.6 where IVC = (2,3), $k = 2$.

Of course, the index k is not known in advance, so linear interpolation is used in block 3, page 2, to estimate the values of α^* and k. The Jacobian for (32), J, is

$$J = \left[\begin{array}{c|c} B & s \\ \hline w & t \end{array}\right]$$

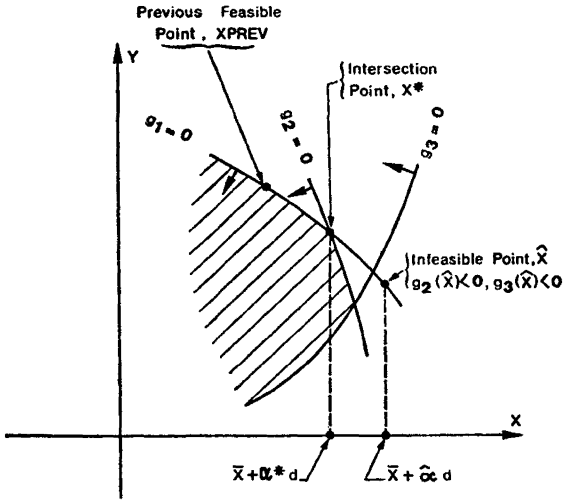


Figure 3.6
Finding First Violated Constraint

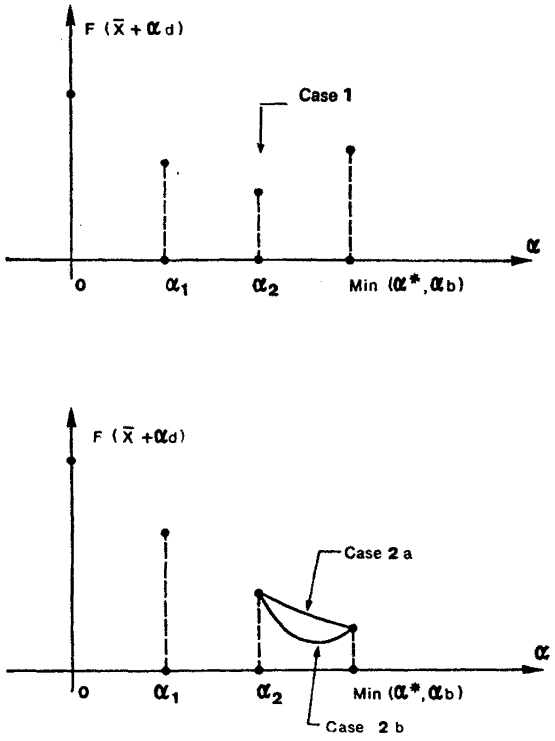


Figure 3.7
Behavior of Reduced Objective at Intersection with New Constraint

where

$$w = \partial g_k / \partial y$$

$$t = (\partial g_k / \partial x)^T d$$

and s is an NB component column vector whose elements are $(\partial g_i/\partial x)^T d$ for $i \in$ IBC. Since J involves only adding a border to the current basis matrix, B, its inverse is easily computed if B^{-1} is known. In our program, the border vectors w, s, t are evaluated at the last feasible point, XPREV, and, as a first try, the current B^{-1} is used, even if it was evaluated at a point other than XPREV. The resulting J^{-1} may be a kind of «hybrid», but if Newton converges, an inversion of B has been saved. Looking at figure 3.6, since Newton converged to $\hat{X}$, using the current B^{-1} and starting from XPREV, one would expect it to converge to the point X^*, which is closer to XPREV, even though an additional constraint has been added. If Newton fails to converge, the same three alternatives as before (with minor modifications, see block 5, page 3) are tried, until convergence is achieved. Then the remaining constraints in IVC are checked to see if they are positive. If one or more are negative, then the linear interpolation estimate of which constraint was violated first was in error. We go back to the linear interpolation block, as the first step toward finding a new value of α^*. This cycle may be repeated a number of times, but the sequence of α^* values should decrease. If not, the procedure is not working and we stop with an error message. Otherwise, we leave this section of code with k as the index of a new binding constraint and α^* as the α value at which this constraint is binding.

The last major task in REDOBJ is to deal with any basic variables which have violated their bounds. This occurs in subroutine BSCHNG (block 4, page 2 of REDOBJ flow chart). Turning to its flow chart, the first action (block 1, page 1) is to check if any bounds on basic variables have been violated. Note that, if any g_i constraints had been violated, the basic variables are now equal to their values at the point $X^* = (y^*, \bar{x} + \alpha^* d)$. For example, in figure 3.6, the y value at X^* might be below its lower bound. If some basic variables do violate their bounds, we proceed through essentially the same logic as is used in dealing with violated g_i constraints. The result is a value of α, α_b, such that all components of $y(\alpha_b)$ except one are strictly between their bounds, with that one, X_l, equal to one of its bound values. Then, after storing the current feasible point as XPREV, we leave subroutine REDOBJ.

The next subroutine encountered in evaluating the reduced objective is NEWG. If a new g_i constraint and/or a bound on a basic variable has been made binding in REDOBJ, NEWG decides whether or not it should remain binding. The behavior of the objective $F(x)$ is the determining factor. Using figure 3.6 as an example, if the one dimensional minimum of F occurs for α less than α^*, then the new constraint g_2 is not made binding. The one dimensional minimum lies in the current constraint intersection, so the step size is reduced,

and the search for an optimal α continues. If, however, F is still decreasing at α^*, the one dimensional minimum is on the boundary of the current constraint intersection. The new constraint g_2 is made binding and the one dimensional search terminates. If both a bound on a basic variable and a new g_i constraint become binding in REDOBJ, this logic still applies, but with α^* replaced by min (α^*, α_b).

The program has been designed to make these decisions without using derivatives. If in block 1, page 1, the objective at min (α^*, α_b) is larger than that at the last feasible point, then the new constraint is not added. This is case 1 of figure 3.7. If the objective is smaller, then we must determine if case 2a or case 2b of figure 3.7 holds. The reduced objective is evaluated at a point nine-tenths of the distance between the last feasible α value and min (α^*, α_b). If the objective value there is smaller than that at min (α^*, α_b), case 2b is assumed to hold and the new constraint is not added. Otherwise case 2 a holds. Either a new constraint is added or the basis is changed, after which we return to the start of the main GRG program with a new reduced problem to solve.

The new constraint incorporation or basis change is carried out in subroutine CONSBS. The input to this subroutine is a list of indices of variables, the candidate list. In block 2 of the NEWG flow chart, this list is set equal to the current list of basic variables. Then CONSBS is called. The outputs of CONSBS are (a) a new list of binding constraint indices (b) a new list of basic variable indices and (c) a new basis inverse, called BINV in the flow chart of CONSBS. On page 1 of this flow chart, the array IREM contains the list of rows which remain to be pivoted in. This is initialized in block 1. The subroutine operates in 2 modes, indicated by the variable MODE. When MODE = 1, CONSBS will choose pivot columns from whatever candidate list was input to it. If a basis inverse could not be constructed from columns in this candidate list, or if the original candidate list included all variables (NCAND = N, block 2), MODE is set to 2, and CONSBS will choose pivot columns from the list of all admissable columns. A column is admissable if its variable is farther than EPSBOUNDS (currently 10^{-6}) from its nearest bound, and if it has not yet been pivoted in.

The main loop of CONSBS begins at block 3. A pivot row is chosen as IROW in block 4. The choice of ISV in block 5 is motivated by the desire to have basic variables as far from their bounds as possible, so that fewer basis changes will be required. The other criterion influencing the choice of basic variables is that the basis matrix should be well-conditioned. We try to insure this by choosing as a prospective pivot column that index, I, in ISV yielding

$$\max_{I \in \text{ISV}} |\text{TAB}(\text{IROW}, I)|$$

This is done in block 6. If the element chosen passes 2 tests we pivot on it (block 8), transforming the Jacobian and entering that column into B^{-1}. The

column pivoted in is marked inadmissable (block 7), and the procedure is repeated for each binding constraint until either B^{-1} has been constructed (N branch, block 9) or the candidate list has been exhausted (Y branch, block 10).

The two tests that a pivot element must pass are (a) its absolute value must be larger than EPSPIVOT (currently 10^{-6}) and (b) the absolute value of the ratio of all other elements in the pivot column to the pivot element must be less than RTOL (currently 100). The first test insures that we do not pivot on an element that is essentially zero, while the second protects against the generation of elements of large absolute value in B^{-1}. Such values are symptomatic of ill-conditioned basis matrices. If either test is failed while MODE = 1, we simply do not pivot in the current row, and move on to the next one.

If, when mode 1 terminates, B^{-1} has not yet been constructed, we attempt to complete its construction by considering columns not in the original candidate list. Mode 2 is entered at marker 5 of the flow chart. In block 11, the candidate list, ICAND, is reset to the set of all remaining admissable columns, MODE is set to 2, and we return to the start of the main iterative loop. If, in this second phase, a pivot element fails the absolute value test, we temporarily mark all columns in ISV inadmissable (by setting their indicator in the IGNORE array to 1, in block 12) and choose a new ISV array. If all admissable matrix elements in a row fail the absolute value test, the matrix is considered to be singular. If a pivot element fails the ratio test in mode 2, it is deleted from ISV in block 13, and the ISV element with second largest absolute value is tried, and so on. If all fail, we set ISKIP = 1 in block 14, which causes us to pivot on the element originally selected, ignoring the ratio test.

4. A NUMERICAL EXAMPLE

Consider the problem

minimize $F(X) = (X_1 - 1)^2 + (X_2 - 0.8)^2$

subject to

$$g_1(X) = X_1 - X_2 - X_3 = 0$$

$$g_2(X) = -X_1^2 + X_2 - X_4 = 0$$

$$g_3(X) = X_1 + X_2 - X_5 - 1 = 0$$

$$X_1 \geqslant 0 \quad , \quad 0 \leqslant X_2 \leqslant 0.8 \quad , \quad X_3 \geqslant 0 \quad , \quad X_4 \geqslant 0 \quad , \quad X_5 \geqslant 0$$

where X_3, X_4, X_5 are slack variables. The feasible region for this problem is graphed in figure 4.1, along with contours of constant value of the objective, The constrained optimum is on the surface $G_2 = 0$ at the point $X_1 = 0.894$, $X_2 = 0.8$. The starting point is $X_1 = 0.6$, $X_2 = 0.4$, which lies on the line $g_3 = 0$, and X_2 is the initial basic variable.

Hence

$$y = X_2, \quad x = (x_1, x_2) = (X_1, X_5), \quad B^{-1} = 1$$

The objective of the first reduced problem, $F(x)$, is obtained by solving g_3 for X_2, yielding

$$X_2 = 1 + X_5 - X_1$$

and substituting this into f, yielding

$$F(x) = (X_1 - 1)^2 + (0.2 + X_5 - X_1)^2$$

whose gradient at $x = (0.6, 0)$ is

$$\nabla F(x) = (\partial F/\partial x_1, \quad \partial F/\partial x_2) = (0, -0.8)$$

Since X_5 is at lower bound, the initial H matrix is (see block 1, page 1, main GRG program flow chart)

$$H = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

so

$$d = -H \nabla F = (0,0)$$

In block 2, page 1 of the main GRG program, we check if X_5 should be released from its lower bound. Since the Kuhn-Tucker multiplier for this bound is negative :

$$\lambda_2^l = \partial F/\partial x_2 = -0.8$$

$x_2 \equiv X_5$ is released from its lower bound. The new H is

$$H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

so now

$$d = -H \nabla F = (0, 0.8)$$

and we enter the one-dimensional search subroutine with $\bar{\alpha} = +\infty$.

This initial one dimensional search varies X_1 and X_5 according to

$$X_1 = 0.6$$
$$X_5 = 0 + 0.8\alpha$$

so we move along the vertical line shown in figure 4.1. The initial step size is chosen to limit the percent change in any variable to less than 5% or, if a variable is zero, to limit the absolute change in such variables to be less than 0.1.

This latter criterion applies here, since only X_2 is changing, and its initial value is zero. Hence, the initial α, α_0, is chosen such that

$$0.8\alpha_0 = 0.1$$

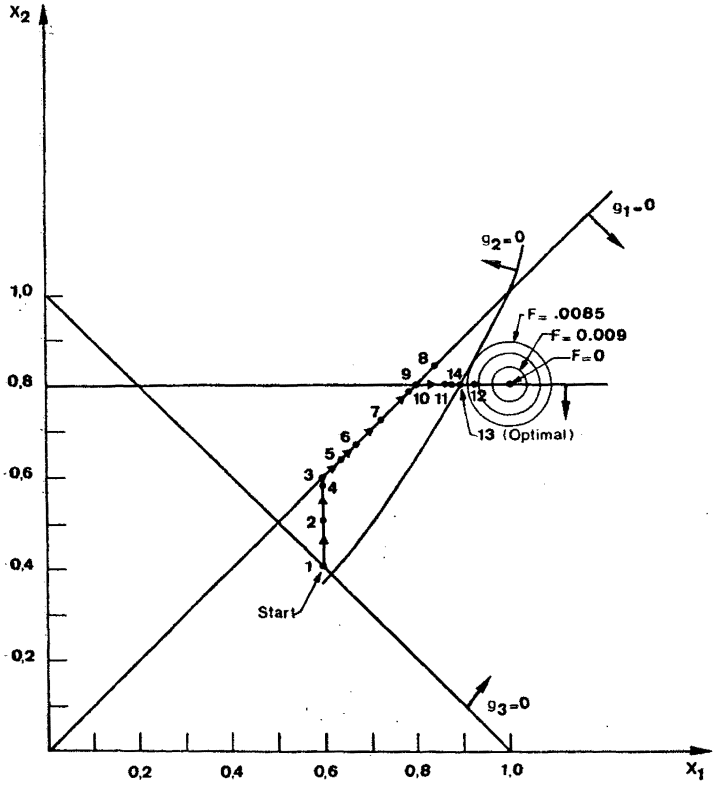


Figure 4.1
Example Problem

or

$$\alpha_0 = 0.125$$

The sequence of α and objective values generated by the one dimensional search is :

α	objective
0	0.32
0.125	0.25
0.250	0.20

The sequence of points generated is shown as points 1, 2, 3 in figure 4.1. For $\alpha = 0.250$, $X_1 = X_2 = 0.6$, which lies on the constraint $g_1 = 0$. The fact

that g_1 has become binding is detected in block 6, page 2 of REDOBJ. Since the reduced objective is lower at $\alpha = 0.250$ than at the previous point, the N branch is taken in block 1 of NEWG. The reduced objective is computed at $\alpha = 0.2375$, and since its value there, 0.2041, is larger than the value at $\alpha = 0.250$, the one dimensional minimum over the interval [0, 0.250] is assumed to occur at $\alpha = 0.250$, i.e. case 2a of figure 3.7 holds. A new basis is constructed in subroutine CONSBS, which yields

$$\text{IBC} = \{ 1 \}$$
$$\text{basic variable } y = X_2$$
$$\text{nonbasic variables } x = (x_1, x_2) = (X_1, X_3)$$
$$B^{-1} = (-1)$$

Control is then transferred back to point (5), page 1, of the main GRG flow chart, and the second major iteration begins.

Since X_3 is at lower bound, H is set to

$$H = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

The reduced gradient at $x = (0.6, 0)$ is

$$\nabla F = (-1.2, 0.4)$$

so

$$d = -H \nabla F = (1.2, 0)$$

No variables are released from their bounds, $\bar{\alpha} = +\infty$, and the one dimensional search begins, varying the nonbasics according to

$$X_1 = 0.6 + 1.2\alpha$$
$$X_3 = 0$$

This corresponds to motion along the line $g_1 = 0$. The sequence of α and objective values generated is

α	objective
0	0.2
0.025	0.1658
0.05	0.1352
0.10	0.0848
0.20	$X_2 < 0.8$ violated

Figure 4.1 shows the corresponding values of X_1 and X_2 as points 5 thru 8. The last α value, $\alpha = 0.20$, yields $X_1 = 0.84$, $X_2 = 0.84$, which violates the upper bound on X_2. This is detected in REDOBJ by subroutine BSCHNG, which attempts to satisfy the upper bound by solving the system

$$g_1(y, x + \alpha d) = 0.6 + 1.2\alpha - X_2 = 0$$
$$X_2 = 0.8$$

which yields $\alpha = 0.166$, corresponding to point 9 of figure 4.1. The objective value at point 9 is 0.0400, which is smaller than the value of .0848 at the last feasible point, point 7. Hence, in NEWG, we take the N branch in block 1, and evaluate F at $\alpha = 0.160$ in the next block, (point 10, figure 4.1) yielding $F = \hat{F} = 0.0433$. This is larger than the value at point 9, so 9 is accepted as the minimum, and subroutine CONSBS is called. This yields a new set of basic variables, and a new basis as follows :

IBC $= \{ 1 \}$
basic variables $y = X_1$
nonbasic variables $x = (X_2, X_3)$
$B^{-1} = (1)$

After leaving CONSBS, we return to page 1 of the main GRG flow chart to begin the third major iteration.

Since both nonbasic variables are at upper bounds, H is set to the zero matrix. To obtain the current reduced objective, we solve the binding constraint for the basic variable in terms of the nonbasic :

$$g_1(X) = X_1 - X_2 - X_3 = 0$$

so

$$X_1 = X_2 + X_3$$

Substituting the above into the objective yields the reduced objective as

$$F(x) = (X_2 + X_3 - 1)^2 + (X_2 - 0.8)^2$$

whose gradient at $X_2 = 0.8$, $X_3 = 0$ is

$$\nabla F = (-0.4, -0.4)$$

In block 2, page 1 of the main GRG program, X_3 has a negative multiplier with value -0.4, so it is released from its lower bound.

H is set to

$$H = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

and the search direction is

$$d = -H \nabla F = (0, 0.4)$$

We begin the one dimensional search, with the nonbasics varied according to

$$X_2 = 0.8$$
$$X_3 = 0 + (0.4)\alpha$$

The α and objective values generated by this search are

α	objective
0	0.04
0.16	0.018
0.32	g_2 violated

The corresponding values of X_1 and X_2 are shown as points 11 and 12 in figure 4.1. Subroutine REDOBJ detects the violation of g_2, and attempts to make g_2 binding by solving the system

$$g_1(y, x + \alpha d) = X_1 - 0.8 - 0.4\alpha = 0$$
$$g_2(y, x + \alpha d) = -X_1^2 + 0.8 = 0$$

This is accomplished in one iteration of Newton's method, starting from initial values

$$(X_1, \alpha) = (0.893, 0.234)$$

(computed in block 3, page 2 of REDOBJ), with inverse Jacobian

$$J^{-1} = \begin{bmatrix} 1 & -0.4 \\ -1.728 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & -0.578 \\ -2.50 & -1.44 \end{bmatrix}$$

The final values are $X_1 = 0.894$, $\alpha = 0.236$ which corresponds to point 13 of figure 4.1. The objective value at that point is 0.0111, which is lower than the value at the last feasible point, point 11. Hence, in NEWG, the abjective is evaluated at point 14, which is 0.9 of the distance between points 11 and 13, corresponding to $\alpha = 0.2285$. The value there is $\hat{F} = 0.0117$, which is larger than the value at point 13, so subroutine CONSBS is called to construct a new basis. The results are

IBC = {2}
basic variables $y = X_1$
nonbasic variables $x = (X_2, X_4)$
$B^{-1} = -0.5590$

Returning to page 1 of the main GRG program flow chart, the reduced gradient is

$$\nabla F = (-0.118, 0.118)$$

Since X_2 is at upper bound and X_4 at lower bound, the Kuhn-Tucker multipliers for these bounds are

$$\lambda_1^u = \partial F/\partial X_2 = 0.118$$

$$\lambda_2^l = \partial F/\partial X_4 = 0.118$$

Since both are positive, point 13 is optimal.

REFERENCES

[1] DANTZIG G. B., *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J., 1963, pp. 94-95.

[2] LUENBERGER D., *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, 1973.

[3] ABADIE J. and CARPENTIER J., « Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints » in *Optimization*, R. Fletcher, Ed., Academic Press, 1969, pp. 37-47.

[4] ABADIE J., « Application of the GRG Algorithm to Optimal Control Problems » in *Nonlinear and Integer Programming*, J. Abadie, Ed., North Holland Publishing Co., 1972, pp. 191-211.

[5] GOLDFARB D., *Extension of Davidons Variable Metric Method to Maximization Under Linear Inequality and Equality Constraints*, SIAM J. Appl. Math, Vol. 17, No. 4, July 1969.

[6] LASDON L. S., FOX R., TAMIR A. and RATNER M., *An Efficient One Dimensional Search Procedure*, Report No. 54, Division of Solid Mechanics, Structures, and Mechanical Design, Case Western Reserve University, June 1973.

[7] HUCKFELDT V., *Nonlinear Optimization with Upper and Lower Bounds*, Technical Memorandum No. 194, Operations Research Department, Case Western Reserve University, Cleveland, Ohio, April 1971.

TABLEAU I

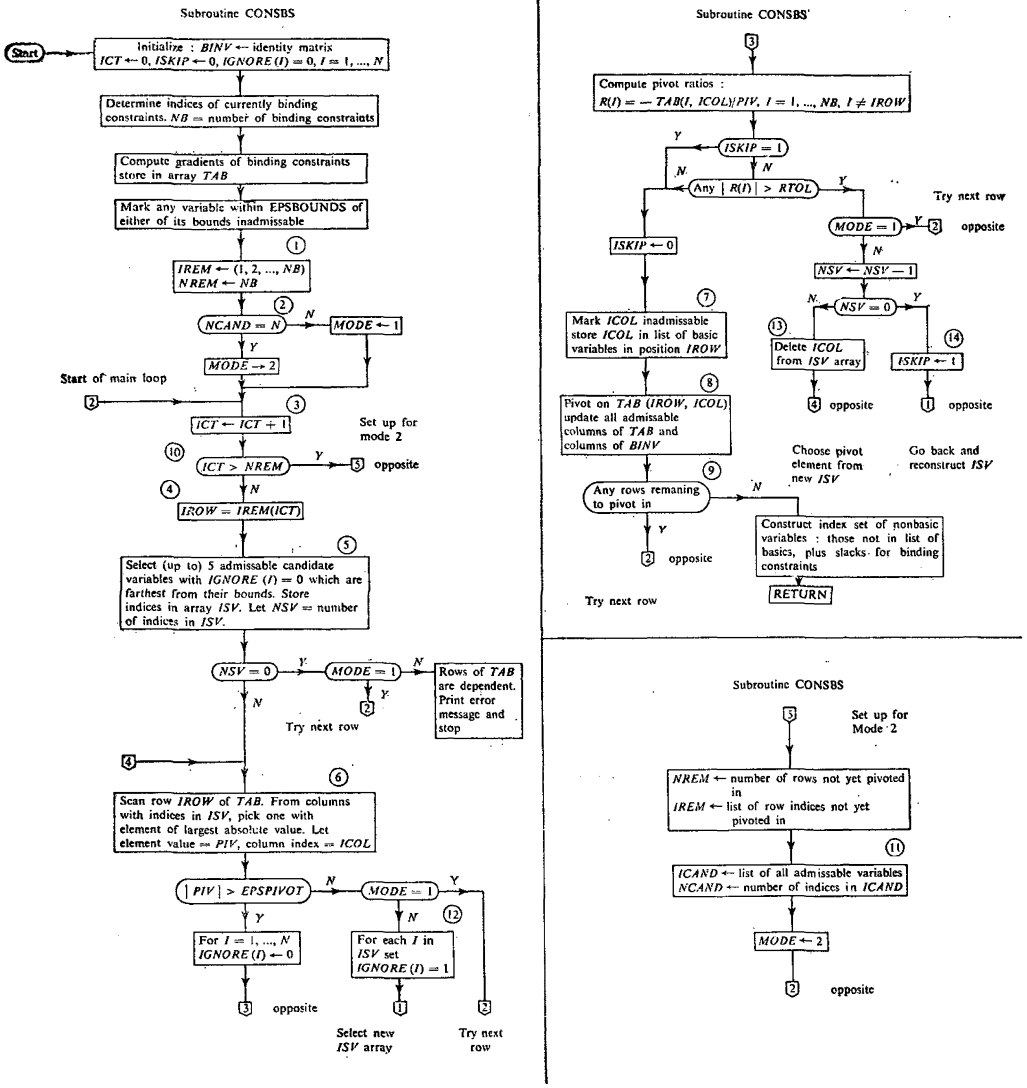

Subroutine CONSBS
Start
Initialize : BINV ← identity matrix
ICT ← 0, ISKIP ← 0, IGNORE (I) = 0, I = 1, ..., N
Determine indices of currently binding constraints. NB = number of binding constraints
Compute gradients of binding constraints store in array TAB
Mark any variable within EPSBOUNDS of either of its bounds inadmissable
IREM ← (1, 2, ..., NB)
NREM ← NB
NCAND = N
MODE ← 1
MODE ← 2
Start of main loop
ICT ← ICT + 1
Set up for mode 2
ICT > NREM
opposite
IROW = IREM(ICT)
Select (up to) 5 admissable candidate variables with IGNORE (I) = 0 which are farthest from their bounds. Store indices in array ISV. Let NSV = number of indices in ISV.
NSV = 0
MODE = 1
Rows of TAB are dependent. Print error message and stop
Try next row
Scan row IROW of TAB. From columns with indices in ISV, pick one with element of largest absolute value. Let element value = PIV, column index = ICOL
| PIV | > EPSPIVOT
MODE = 1
For I = 1, ..., N IGNORE (I) ← 0
For each I in ISV set IGNORE (I) = 1
opposite
Select new ISV array
Try next row
Subroutine CONSBS
Compute pivot ratios : R(I) = — TAB(I, ICOL)/PIV, I = 1, ..., NB, I ≠ IROW
ISKIP = 1
Any | R(I) | > RTOL
Try next row
MODE = 1
opposite
ISKIP ← 0
NSV ← NSV — 1
NSV = 0
Mark ICOL inadmissable store ICOL in list of basic variables in position IROW
Delete ICOL from ISV array
ISKIP ← 1
opposite
opposite
Pivot on TAB (IROW, ICOL) update all admissable columns of TAB and columns of BINV
Choose pivot element from new ISV
Go back and reconstruct ISV
Any rows remaning to pivot in
Construct index set of nonbasic variables : those not in list of basics, plus slacks for binding constraints
RETURN
opposite
Try next row
Subroutine CONSBS
Set up for Mode 2
NREM ← number of rows not yet pivoted in
IREM ← list of row indices not yet pivoted in
ICAND ← list of all admissable variables
NCAND ← number of indices in ICAND
MODE ← 2
opposite

TABLEAU II

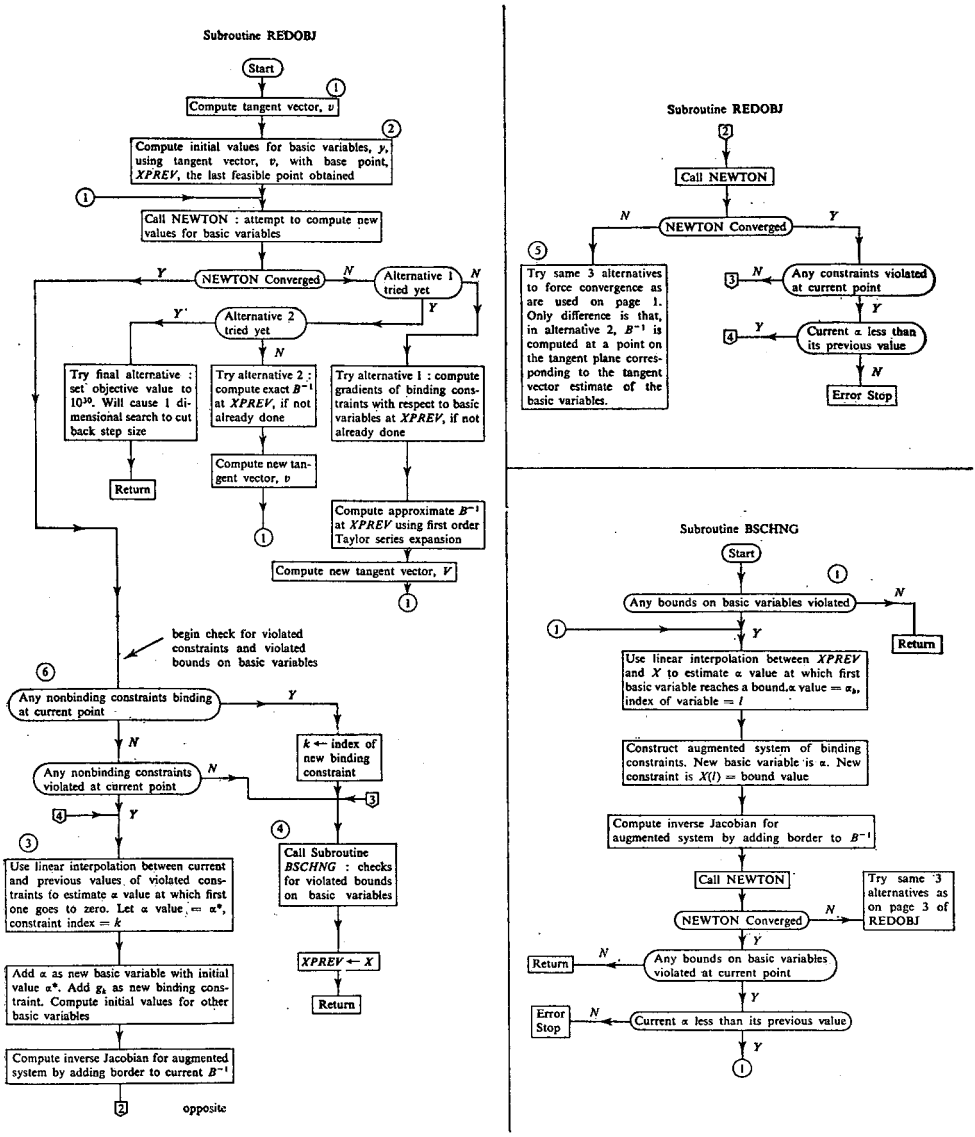

Subroutine REDOBJ
Start
Compute tangent vector, v
Compute initial values for basic variables, y, using tangent vector, v, with base point, XPREV, the last feasible point obtained
Call NEWTON : attempt to compute new values for basic variables
NEWTON Converged
Alternative 1 tried yet
Alternative 2 tried yet
Try final alternative : set objective value to 10^30. Will cause 1 dimensional search to cut back step size
Return
Try alternative 2 : compute exact B^-1 at XPREV, if not already done
Compute new tangent vector, v
Try alternative 1 : compute gradients of binding constraints with respect to basic variables at XPREV, if not already done
Compute approximate B^-1 at XPREV using first order Taylor series expansion
Compute new tangent vector, V
begin check for violated constraints and violated bounds on basic variables
Any nonbinding constraints binding at current point
Any nonbinding constraints violated at current point
k ← index of new binding constraint
Use linear interpolation between current and previous values of violated constraints to estimate α value at which first one goes to zero. Let α value = α*, constraint index = k
Call Subroutine BSCHNG : checks for violated bounds on basic variables
XPREV ← X
Return
Add α as new basic variable with initial value α*. Add g_k as new binding constraint. Compute initial values for other basic variables
Compute inverse Jacobian for augmented system by adding border to current B^-1
opposite
Subroutine REDOBJ
Call NEWTON
NEWTON Converged
Try same 3 alternatives to force convergence as are used on page 1. Only difference is that, in alternative 2, B^-1 is computed at a point on the tangent plane corresponding to the tangent vector estimate of the basic variables.
Any constraints violated at current point
Current α less than its previous value
Error Stop
Subroutine BSCHNG
Start
Any bounds on basic variables violated
Return
Use linear interpolation between XPREV and X to estimate α value at which first basic variable reaches a bound, α value = α_s, index of variable = l
Construct augmented system of binding constraints. New basic variable is α. New constraint is X(l) = bound value
Compute inverse Jacobian for augmented system by adding border to B^-1
Call NEWTON
NEWTON Converged
Try same 3 alternatives as on page 3 of REDOBJ
Return
Any bounds on basic variables violated at current point
Error Stop
Current α less than its previous value

TABLEAU III

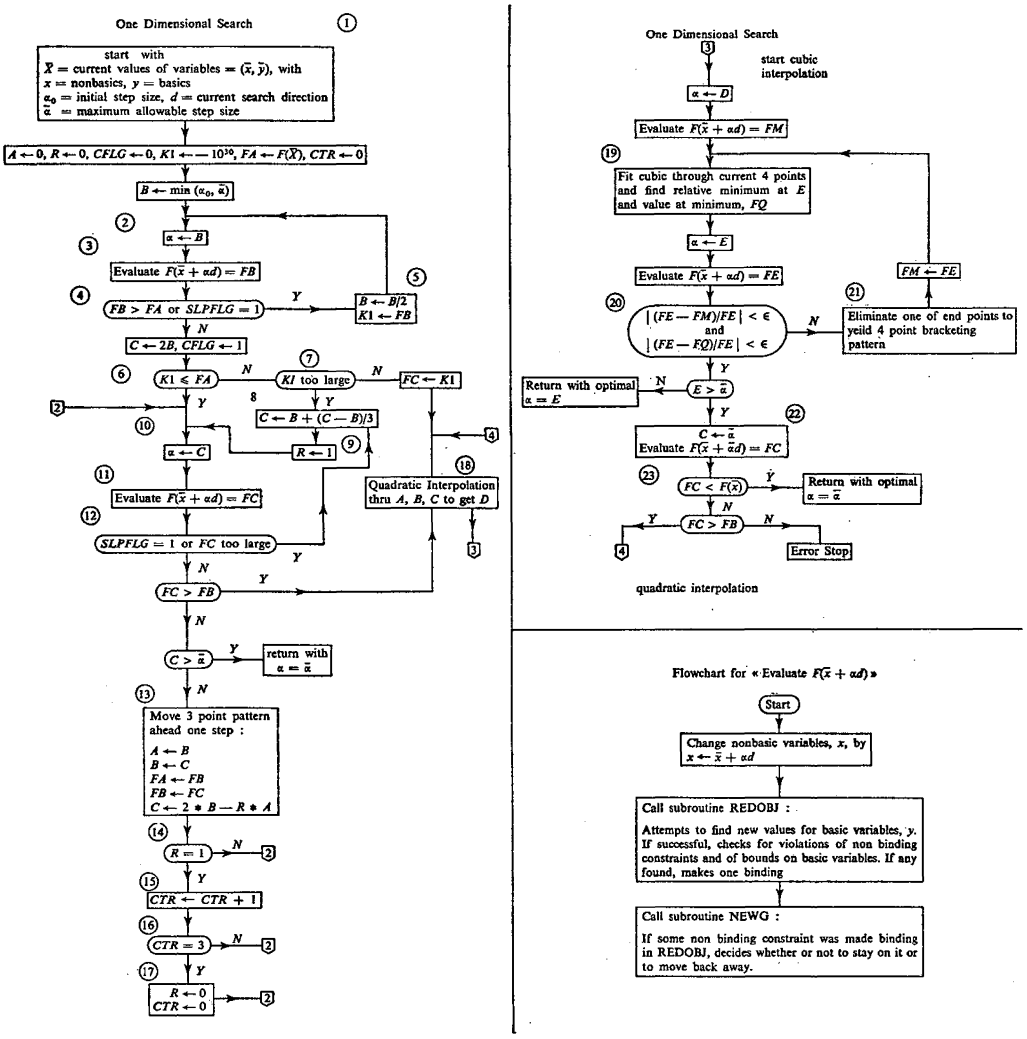
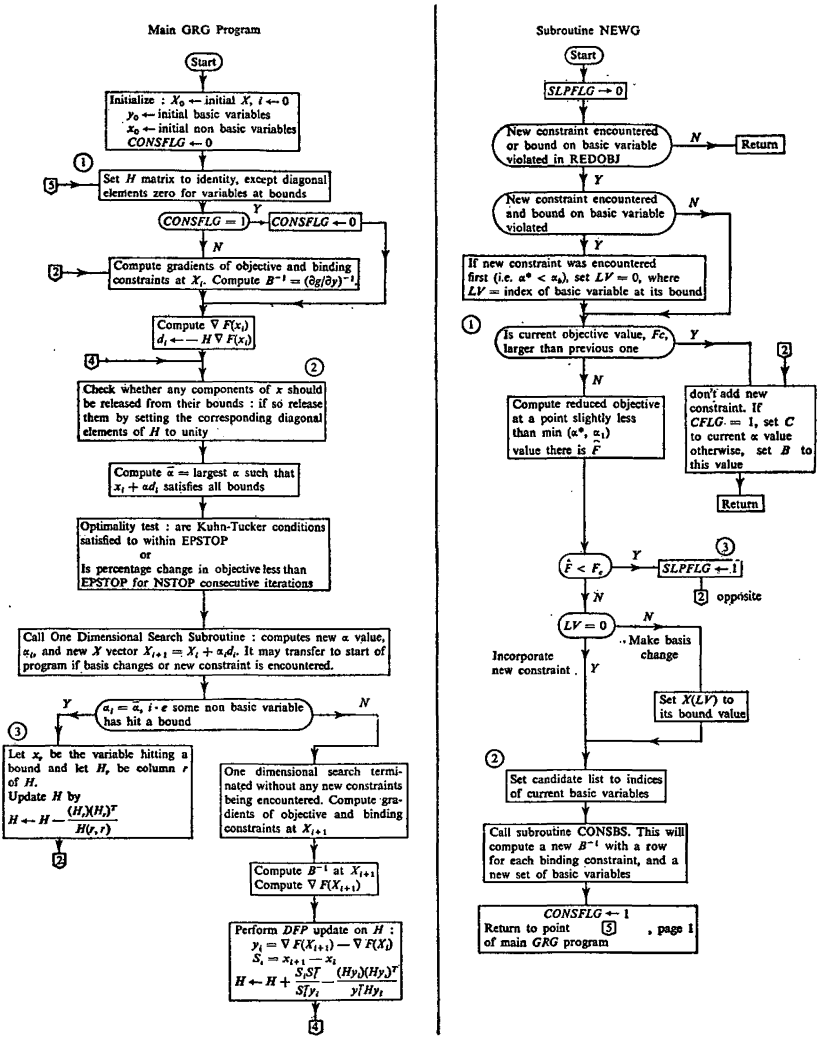

One Dimensional Search
start with
$\bar{X}$ = current values of variables = $(\bar{x}, \bar{y})$, with
x = nonbasics, y = basics
α_0 = initial step size, d = current search direction
$\bar{\alpha}$ = maximum allowable step size
$A \leftarrow 0, R \leftarrow 0, CFLG \leftarrow 0, K1 \leftarrow -10^{30}, FA \leftarrow F(\bar{X}), CTR \leftarrow 0$
$B \leftarrow \min(\alpha_0, \bar{\alpha})$
$\alpha \leftarrow B$
Evaluate $F(\bar{x} + \alpha d) = FB$
$FB > FA$ or $SLPFLG = 1$
$B \leftarrow B/2$
$K1 \leftarrow FB$
$C \leftarrow 2B, CFLG \leftarrow 1$
$K1 \leqslant FA$
$K1$ too large
$FC \leftarrow K1$
$C \leftarrow B + (C - B)/3$
$R \leftarrow 1$
$\alpha \leftarrow C$
Evaluate $F(\bar{x} + \alpha d) = FC$
$SLPFLG = 1$ or FC too large
$FC > FB$
$C > \bar{\alpha}$
return with $\alpha = \bar{\alpha}$
Move 3 point pattern ahead one step :
$A \leftarrow B$
$B \leftarrow C$
$FA \leftarrow FB$
$FB \leftarrow FC$
$C \leftarrow 2 * B - R * A$
$R = 1$
$CTR \leftarrow CTR + 1$
$CTR = 3$
$R \leftarrow 0$
$CTR \leftarrow 0$
Quadratic Interpolation thru A, B, C to get D
One Dimensional Search
start cubic interpolation
$\alpha \leftarrow D$
Evaluate $F(\bar{x} + \alpha d) = FM$
Fit cubic through current 4 points and find relative minimum at E and value at minimum, FQ
$\alpha \leftarrow E$
Evaluate $F(\bar{x} + \alpha d) = FE$
$|(FE - FM)/FE| < \epsilon$ and $|(FE - FQ)/FE| < \epsilon$
$FM \leftarrow FE$
Eliminate one of end points to yeild 4 point bracketing pattern
Return with optimal $\alpha = E$
$E > \bar{\alpha}$
$C \leftarrow \bar{\alpha}$
Evaluate $F(\bar{x} + \bar{\alpha} d) = FC$
$FC < F(\bar{x})$
Return with optimal $\alpha = \bar{\alpha}$
$FC > FB$
Error Stop
quadratic interpolation
Flowchart for « Evaluate $F(\bar{x} + \alpha d)$ »
Start
Change nonbasic variables, x, by $x \leftarrow \bar{x} + \alpha d$
Call subroutine REDOBJ :
Attempts to find new values for basic variables, y. If successful, checks for violations of non binding constraints and of bounds on basic variables. If any found, makes one binding
Call subroutine NEWG :
If some non binding constraint was made binding in REDOBJ, decides whether or not to stay on it or to move back away.

TABLEAU IV


Main GRG Program
Start
Initialize : $X_0 \leftarrow$ initial X, $i \leftarrow 0$
$y_0 \leftarrow$ initial basic variables
$x_0 \leftarrow$ initial non basic variables
$CONSFLG \leftarrow 0$
Set H matrix to identity, except diagonal elements zero for variables at bounds
$CONSFLG = 1$
Y
$CONSFLG \leftarrow 0$
N
Compute gradients of objective and binding constraints at X_i. Compute $B^{-1} = (\partial g/\partial y)^{-1}$
Compute $\nabla F(x_i)$
$d_i \leftarrow -H \nabla F(x_i)$
Check whether any components of x should be released from their bounds : if so release them by setting the corresponding diagonal elements of H to unity
Compute $\bar{\alpha} =$ largest α such that $x_i + \alpha d_i$ satisfies all bounds
Optimality test : are Kuhn-Tucker conditions satisfied to within EPSTOP
or
Is percentage change in objective less than EPSTOP for NSTOP consecutive iterations
Call One Dimensional Search Subroutine : computes new α value, α_i, and new X vector $X_{i+1} = X_i + \alpha_i d_i$. It may transfer to start of program if basis changes or new constraint is encountered.
$\alpha_i = \bar{\alpha}$, i.e. some non basic variable has hit a bound
Y
N
Let x_r be the variable hitting a bound and let H_r be column r of H.
Update H by
$H \leftarrow H - \frac{(H_r)(H_r)^T}{H(r,r)}$
One dimensional search terminated without any new constraints being encountered. Compute gradients of objective and binding constraints at X_{i+1}
Compute B^{-1} at X_{i+1}
Compute $\nabla F(X_{i+1})$
Perform DFP update on H :
$y_i = \nabla F(X_{i+1}) - \nabla F(X_i)$
$S_i = x_{i+1} - x_i$
$H \leftarrow H + \frac{S_i S_i^T}{S_i^T y_i} - \frac{(Hy_i)(Hy_i)^T}{y_i^T H y_i}$
Subroutine NEWG
Start
$SLPFLG \rightarrow 0$
New constraint encountered or bound on basic variable violated in REDOBJ
N
Return
Y
New constraint encountered and bound on basic variable violated
N
Y
If new constraint was encountered first (i.e. $\alpha^* < \alpha_1$), set $LV =$ index of basic variable at its bound
Is current objective value, Fc, larger than previous one
Y
2
N
Compute reduced objective at a point slightly less than min (α^*, α_1) value there is $\hat{F}$
don't add new constraint. If $CFLG = 1$, set C to current α value otherwise, set B to this value
Return
$\hat{F} < F_c$
Y
3
$SLPFLG \leftarrow 1$
2 opposite
N
$LV = 0$
N
Make basis change
Incorporate new constraint.
Y
Set $X(LV)$ to its bound value
2
Set candidate list to indices of current basic variables
Call subroutine CONSBS. This will compute a new B^{-1} with a row for each binding constraint, and a new set of basic variables
$CONSFLG \leftarrow 1$
Return to point 5 , page 1 of main GRG program