AN ACTIVE-CONSTRAINT LOGIC FOR NONLINEAR PROGRAMMING

by

Alok Das

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Aerospace and Ocean Engineering

APPROVED:

H. J. Kelley, Chairman

R. T. Haftka

E. M. Cliff

F. H. Lutze

J. A. Burns

May, 1982

Blacksburg, Virginia

ACKNOWLEDGMENT

The author wishes to express his sincere appreciation to Dr. Henry J. Kelley for the guidance and encouragement extended throughout this work. Special thanks are due to Dr. E. M. Cliff for a number of valuable suggestions. The author would also like to thank Dr. R. T. Haftka, Dr. F. H. Lutze and Dr. J. A. Burns for reviewing the thesis and providing suggestions.

TABLE OF CONTENTS

LIST OF TABLES

SYMBOLS

A	n x M matrix of constraint gradients for linear constraints
A	n x m matrix of constraint gradients for linear constraints
A_1	1 x 1 Grammian matrix of constraint gradients
c	Constant
d	M-vector used for defining linear constraints
f(x)	Objective function
g(x)	M-vector of constraint functions
H	n x n positive definate symmetric matrix
m	Number of constraints satisfied as equalities
M	Total number of constraints
n	Number of variables
p	Number of constraints satisfied as equalities but considered inactive
$P^1[-f_x]$	Projection of $-f_x$ on the tangent subspace of 1 of the constraints satisfied as equalities
q	n-vector
Q	n x n positive definate symmetric matrix
r	Number of constraints taken as active
s	n-vector giving the step
x	n-vector of variables
λ	m-vector of projection multipliers when m constraints are taken as active

$\bar{\lambda}$	m-vector of projection multipliers when each constraint is considered exclusive of others
$\tilde{\lambda}$	r-vector of projection multipliers when r ($\leq$ m) constraints are taken as active
$\lambda_{i,j}$	Projection multiplier corresponding to the i^{th} constraint with the j^{th} constraint dropped from the active set
	Subscripts
i,j,k	i^{th}, j^{th}, k^{th} component of the vector respectivelly
x	Derivative with respect to x

1. INTRODUCTION

There is considerable spread in opinion and in practice on the basis for the choice of active constraints in nonlinear-programming processes. With some approaches, inequalities are added or dropped from the active set only when absolutely necessary for continued progress, while with others the active constraint set is determined anew each cycle. With the least restrictive schemes of this latter type, cycling back and forth between active and inactive status tends to occur, while with the most restrictive of the former type, reduction in the performance index is impeded (Ref. 1).

In the active-constraint logic suggested by Fletcher for quadratic programming problems (Ref. 2), all the constraints satisfied as equalities are taken as active. Constraints are added as necessary until the minimum of the function, subject to the constraints in the active set satisfied as equalities, is reached. If the Kuhn-Tucker conditions are not met at this point, further reduction in the objective function can be made by deleting from the active set any constraint for which the Lagrange multiplier has the "wrong" sign. If two or more Kuhn-Tucker violators are present, it is not obvious which constraints should be dropped, and in what order. Fletcher deleted only one constraint, that corresponding to the most negative multiplier, the constraints being initially normalized.

Fletcher's method is not very efficient computationally for general linearly-constrained problems. As pointed out by Powell (Ref.

3), a general function has to be minimized subject to linear constraints before a constraint is deleted from the active set. It should be pointed out that by its very nature, Fletcher's method prevents cycling between active and inactive constraint status. It seems advantageous to drop constraints more often, usually at every iteration; however, this could lead to cycling.

The problem of constraints cycling between active and inactive status is an important factor in any active-constraint logic. This matter has been discussed in some detail in Refs. 2 and 12. A crude but effective anticycling rule is by Zoutendijk (Ref. 11). In this, if a constraint dropped earlier became active again, then that constraint is retained in the active set until a minimum is reached with the constraints in the active set satisfied as equalities. Zigzagging could also be avoided by using Rosen's method of dropping constraints (Ref. 10). Here emphasis is placed upon estimation of objective function improvements to be realized dropping each violator singly. A constraint is dropped only if it is estimated that this will result in a sufficient decrease in the function. McCormick (Ref. 13) suggested yet another method to the same cause. Gill and Murray (Ref. 12) recommend Zoutendijk rule along with a fairly slack tolerance within which the objective function is minimized in each active constraint set as an effective measure against zigzagging.

In the least restrictive schemes, constraints are dropped at every iteration such that the remaining constraints satisfy Kuhn-Tucker conditions. In the presence of multiple Kuhn-Tucker violators, it is not clear how the constraints are to be dropped. Sargent and Murtagh

(Ref. 8 and 9) recommend dropping the Kuhn-Tucker violators one at a time, recomputing the multipliers each time, until no negative multipliers remain. They suggest that each time either the constraint with the most negative multiplier or, following Rosen's method, the constraint which is estimated to cause the maximum decrease in the objective function be dropped. As shown in the following (Lemma 1, also in Refs. 3 and 9), a Kuhn-Tucker violator can be safely dropped alone, but there is no guarantee that the resulting step will not violate it if another constraint is also dropped. This problem arises whenever more than one constraint is dropped at any iteration.

With nonlinear constraints, Kuhn-Tucker screening can be done using multipliers evaluated with linear approximations to the constraints (i.e., using the projection multipliers). Kelley, Lefton and Johnson (Ref. 6) applied these conditions in a sequence of calculations, first with all of the candidates included, then successively with Kuhn-Tucker violators dropped, as many times as necessary, until all multiplier components satisfy Kuhn-Tucker or all candidates have been screened out. The difficulty encountered in the presence of multiple violators was alleviated subsequently by a procedure which usually represents an improvement over dropping violators in an arbitrary order and later rechecking them after successive reductions have led to a no-violator constraint set (Ref. 7). The multiplier components for the inequality constraints, each calculated exclusive of other inequalities, are employed. Members of the subset of dual violators, those having the wrong multiplier-component signs both singly and in combination, are dropped out first, one at a time.

This dual-violator rule is valid only for the two-constraint case, as will be seen in the following.

Little research has been done on computational comparison of the various active-set strategies. In an interesting paper Lenard (Ref. 1) compared a "most-constrained" strategy with a "least-constrained" strategy for general linearly constrained problems. Both quadratic and nonquadratic functions were used for the objective function. The least-constrained strategy almost always required fewer iterations than the most-constrained strategy. The difference becomes more apparent as the number of variables increases. In fact, for most problems with more than twelve variables (from among those used in the study), the number of iterations can be approximated as a linear function of the number of variables with slope of one in the least-constrained case and a slope of two in the most-constrained case.

The usefulness of least-constrained strategies will be enhanced significantly if a rule can be found for dropping constraints without the fear that the resulting step may violate some of the dropped constraints. This aspect has been investigated in the present work.

The Kuhn-Tucker conditions are reviewed in the next chapter. A number of results are obtained in Chapter 3. It is essential for at least one Kuhn-Tucker violator to be present in the set of constraints being dropped. This is only a necessary condition for ensuring that none of the dropped constraints are violated by the resulting step. Theorem 2 gives conditions which are sufficient but not necessary for dropping one or more constraints. Using the rule suggested by Theorem 2 for dropping constraints, it is possible to drop more than one constraint without violating them, but there is no guarantee that the resulting active set satisfies Kuhn-Tucker conditions. The desired attributes of the active set of constraints sought have been converted into necessary and sufficient conditions in Lemma 2. Though not very helpful in generating the active set, these conditions serve as a test which any candidate active-constraint logic must pass.

The dual-violator rule is the result of the first systematic attempt in dropping more than one constraint while making sure that the dropped constraints are not violated. Unfortunately, the dual violator rule breaks down if more than two constraints are satisfied as equalities at the point under consideration. The same holds for Sargent's worst-violator rule. This aspect is discussed in detail in Chapter 4. Counterexamples to both of these rules are given in Appendix B to illustrate the pitfalls of oversimplified active-constraint logic.

The three-constraint case is studied in Chapter 5. The problem with three constraints is significantly more complex than the problem with only two constraints. Using the particular structure of this case, a rule for generating the active set of constraints is obtained. An important aspect which has been neglected in the literature is the existence of the desired active set. With the help of Lemmas 3 and 4, it has been shown that the active set exists for problems with two and three constraints. For the general case of m constraints, a combination of Theorem 2 and the rules developed for two and three constraints is recommended for use. This is discussed in Chapter 6.

A computational study was done to assess the effectiveness of the active-set strategy proposed in this study in comparison to some existing strategies. Only quadratic-programming problems were used in this study. The active-set strategies were compared for the effect they had on the number of times the gradient of the objective function was evaluated. The details of the algorithm are discussed in Chapter 7. The active-set strategy proposed in this study almost always performed as well or better than the other strategies.

2. REVIEW OF KUHN-TUCKER CONDITIONS

In many engineering problems, it is necessary to minimize or maximize a real-valued function (f(x)) subject to several inequality constraints, say

$$g_1(x) \leq 0 \quad , \quad g_2(x) \leq 0 \quad , \quad \dots . \quad , \quad g_M(x) \leq 0 \tag{i}$$

where x is an n-vector and each g_j is a real-valued function. The problem with f, g_1, g_2, ..., g_M as linear functions is known as the linear programming problem. Rapid progress in linear programming led to the study of optimization where the f and g_j's are nonlinear. A fundamental theoretical paper in nonlinear programming by Kuhn and Tucker appeared in 1951 (Ref. 17). Since then, nonlinear programming has existed as a distinguishable subject. Two outstanding surveys on the development of nonlinear programming are Takayama (Ref. 20) and Kuhn (Ref. 21).

Consider the problem of minimizing f(x) subject to inequality constraints (i). Let f, g_1, g_2, ..., g_M have continuous partial derivatives in an open set containing $\bar{x}$. If $\bar{x}$ minimizes f(x) subject to (i), then, under certain constraint-qualification conditions, the Kuhn-Tucker theorem associates with each inequality a real number (called a multiplier) λ_i, such that

$$\frac{\partial L(\bar{x})}{\partial x_i} = 0 \quad , \quad i = 1,2,\dots,M \tag{ii}$$

$$\sum_{i=1}^{M} \lambda_i \, g_i(\bar{x}) = 0 \tag{iii}$$

and

$$\lambda_i \geq 0 \quad , \quad i = 1,2,\ldots,M \tag{iv}$$

where the Lagrangian function is given by

$$L(x,\lambda) = f(x) + \sum_{i=1}^{M} \lambda_i \, g_i(x)$$

The constraint-qualification condition used by Kuhn and Tucker is difficult to verify. A much more easily verifiable constraint-qualification condition requires the linear independence of the gradients of the constraint functions, g_i, at $\bar{x}$ (Ref. 22). Another constraint-qualification condition is by Slater.

In an unpublished work, Karush (Ref. 18) had used a technique from the calculus of variations to obtain results similar to the Kuhn-Tucker theorem. Karush's results preceded the classical theorem of Kuhn and Tucker by twelve years. F. John (Ref. 19) had also obtained important results prior to Kuhn and Tucker. He defined the Lagrangian as $L = \lambda_0 \, f(x) + \sum_{i=1}^{M} \lambda_i \, g_i(x)$ and hence did not require any constraint qualification condition. In John's results, λ_0 can be zero. The constraint qualification employed by Karush and Kuhn-Tucker amounts to guaranteeing $\lambda_0 > 0$ and hence that it can be factored out. This condition is also known as a normality condition.

Theorems corresponding to the Kuhn-Tucker and John theorems, discussed above, are available when the differentiability of f and g's are not assumed. Instead it is assumed that the feasible region S is convex and f, g_1, ..., g_M are all convex on S. The Kuhn-Tucker saddle-point theorem can be stated as follows.

If $\bar{x}$ minimizes $f(x)$ subject to (i) then, under Slater's constraint-qualification condition, there exist non-negative multipliers $\bar{\lambda}_i$, $i = 1,\dots,M$ such that $(\bar{x},\bar{\lambda})$ is a saddle point of $L(x,\lambda)$, i.e.,

$$L(\bar{x},\lambda) \leq L(\bar{x},\bar{\lambda}) \leq L(x,\bar{\lambda})$$

and

$$\sum_{i=1}^{M} \bar{\lambda}_i \, g_i(\bar{x}) = 0$$

for all $\lambda \geq 0$ and for all x in S.

The Slater constraint qualification condition simply requires that there exists an x in S (convex) such that the convex functions $g_i(x)$ satisfy $g_i(x) < 0$, $i = 1,2,\dots,M$.

3. GRADIENT PROJECTION

Consider the problem of minimizing a nonlinear function f(x) subject to M nonlinear inequalities

$$g_i(x) \leq 0 \qquad i = 1,\ldots,M \tag{1}$$

where x is an n-vector. Let x be a feasible point at which $m \leq n$ constraints are satisfied as equalities.

The Kuhn-Tucker screening is done via the projection multipliers. These multipliers correspond to the problem of minimizing a linear approximation to f(x) subject to linearized constraints and to a quadratic constraint on step-size. The Kuhn-Tucker conditions employed become identical to those for the original problem at the constrained minimum sought. For the linearized problem subject to a quadratic constraint on step size, the projection multipliers are given by

$$\lambda = -\,(g_x{}^T H\, g_x)^{-1}\, g_x{}^T H\, f_x\ , \tag{2}$$

where H is a positive-definite symmetric matrix

f_x is an n-vector whose ith element is $\partial f/\partial x_i$

and g_x is an n x m matrix whose jth column is g_{jx}, the gradient of g_j with respect to vector x.

The m constraints satisfied as equalities are assumed to be independent, i.e., the m columns of the matrix g_x are linearly independent, a "normality" assumption. Eqn. (2) is derived in Appendix A.

The active-set choice via Kuhn-Tucker screening depends upon the metric H, as mentioned in Ref. 4. To simplify the algebra, take H as the identity matrix. This is equivalent to a linear nonsingular transformation on x. Then

$$\lambda = (g_x^T g_x)^{-1} \bar{\lambda} , \qquad (3)$$

where

$$\bar{\lambda} = - (g_x^T f_x) . \qquad (4)$$

$\bar{\lambda}_i$ is the projection multiplier for a problem with the ith constraint considered exclusive of the others. Without loss of generality, the gradients of the constraints have been normalized here and henceforth, i.e.,

$$g_{i_x}^T g_{i_x} = 1 \quad , \quad i = 1,\ldots,m .$$

This is equivalent to multiplying each constraint function by a (generally different) positive scaling factor.

Generally, all the projection multipliers will not have the 'right' sign (> 0). It is desired to have an active-set strategy which chooses the 'correct' constraint subset such that all the projection multipliers are positive and the projected gradient direction does not violate the linearized versions of any of the dropped constraints.

The remaining sections of this chapter are devoted to proving some theorems and lemmas which aid in generating active constraint logic. As mentioned earlier, the two essential features of a good active-constraint logic are

(i) Constraints in the active set satisfy the Kuhn-Tucker conditions

(ii) The projected gradient direction does not violate any of the constraints which are satisfied as equalities but considered inactive.

Definition: A subset of the set of constraints satisfied as equalities can be "safely dropped" if a step along the negative projected-gradient direction with the remaining constraints taken as active does not violate the linear approximation to any of the constraints in this subset.

Theorem 1. A necessary condition for safely dropping a subset of constraints is the existence of at least one Kuhn-Tucker violator in the set.

Proof: A set of r constraints is to be retained and the remaining p (= m - r) dropped. Since the constraints can be rearranged without any loss of generality, let the first r constraints be retained and the corresponding projection multipliers be $\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_r$. The projection of $-f_x$ on the subspace formed by the intersection of the tangent hyperplanes of the first r constraints $g_i = 0$, $i = 1,2,\ldots,r$, is

$$P^r[-f_x] = - f_x - \sum_{i=1}^{r} \hat{\lambda}_i \, g_{i_x} \tag{5}$$

A step along $P^r[-f_x]$ will violate (not violate) $g_j \leq 0$ if

$$g_{j_x}^T \, P^r[-f_x] > 0 \; (< 0)$$

for $j = r+1, r+2, \ldots, m$

i.e., $$\bar{\lambda}_j - \sum_{i=1}^{r} (g_{j_x}^T \, g_{i_x}) \, \hat{\lambda}_i > 0 \; (< 0) \tag{6}$$

From eqn. (3) $\bar{\lambda} = A_m \lambda$

or
$$\begin{bmatrix} \bar{\Lambda}_r \\ \\ \bar{\Lambda}_p \end{bmatrix} = \begin{bmatrix} A_r & G \\ \\ G^T & A_p \end{bmatrix} \begin{bmatrix} \Lambda_r \\ \\ \Lambda_p \end{bmatrix} \tag{7}$$

where

$\bar{\Lambda}_r = [\bar{\lambda}_1 \quad \bar{\lambda}_2 \quad \cdots \quad \bar{\lambda}_r]^T$

$\bar{\Lambda}_p = [\bar{\lambda}_{r+1} \quad \bar{\lambda}_{r+2} \quad \cdots \quad \bar{\lambda}_m]^T$

$\Lambda_r = [\lambda_1 \quad \lambda_2 \quad \cdots \quad \lambda_r]^T$

$\Lambda_p = [\lambda_{r+1} \quad \lambda_{r+2} \quad \cdots \quad \lambda_m]^T$

$$G = \begin{bmatrix} ({g_1}_x^T \, {g_{r+1}}_x) & ({g_1}_x^T \, {g_{r+2}}_x) & \cdots & ({g_1}_x^T \, {g_m}_x) \\ ({g_2}_x^T \, {g_{r+1}}_x) & ({g_2}_x^T \, {g_{r+2}}_x) & \cdots & ({g_2}_x^T \, {g_m}_x) \\ \vdots & \vdots & & \vdots \\ ({g_r}_x^T \, {g_{r+1}}_x) & ({g_r}_x^T \, {g_{r+2}}_x) & \cdots & ({g_r}_x^T \, {g_m}_x) \end{bmatrix}$$

and

A_r and A_p are the gram matrices for the first r and the remaining p constraints respectively.

The above equation yields

$$\bar{\lambda}_j = \sum_{i=1}^{m} ({g_i}_x^T \, {g_j}_x) \, \lambda_i \quad , \quad j = r+1, r+2, \ldots, m \tag{8}$$

Eqn. (6) becomes

$$\sum_{i=1}^{r} ({g_j}_x^T \, {g_i}_x) \, (\lambda_i - \hat{\lambda}_i) + \sum_{i=r+1}^{m} ({g_j}_x^T \, {g_i}_x) \, \lambda_i > 0 \; (< 0) \tag{9}$$

for $j = r+1, r+2, \ldots, m$.

In matrix form these p equations can be written as

$$G^T \Lambda_r - G^T \hat{\Lambda}_r + A_p \Lambda_p > 0 \; (< 0) \qquad (10)$$

where $\hat{\Lambda}_r = [\hat{\lambda}_1 \quad \hat{\lambda}_2 \quad \ldots \quad \hat{\lambda}_r]^T$

Since

$$\hat{\Lambda}_r = A_r^{-1} \bar{\Lambda}_r$$

and

$$\bar{\Lambda}_r = A_r \Lambda_r + G \Lambda_p$$

eqn. (10) becomes

$$[A_p - G^T A_r^{-1} G] \Lambda_p > 0 \; (< 0) \qquad (11)$$

If the constraints are rearranged so that the last p constraints are made the first p, the gram matrix of the m constraint gradients is

$$B = \begin{bmatrix} A_p & G^T \\ G & A_r \end{bmatrix}$$

Since matrix A_m and hence matrix B is symmetric and positive definite, B^{-1} will also be symmetric and positive definite. Moreover from Ref. 14,

$$B^{-1} = \left[\begin{array}{c|c} (A_p - G^T A_r^{-1} G)^{-1} & - A_p^{-1} G^T (A_r - G A_p^{-1} G^T)^{-1} \\ \hline - A_r^{-1} G(A_p - G^T A_r^{-1} G)^{-1} & (A_r - G A_p^{-1} G^T)^{-1} \end{array}\right]$$

Therefore matrix $(A_p - G^T A_r^{-1} G)$ is positive definite.

Let $$b = [A_p - G^T A_r^{-1} G] \Lambda_p \qquad (12)$$

and suppose that all the elements of $\Lambda_p > 0$.

The p constraints can be safely dropped if all the elements of $b < 0$.

Eqn. (12) gives

$$\Lambda_p^T b = \Lambda_p^T [A_p - G^T A_r^{-1} G] \Lambda_p > 0 \qquad (13)$$

So if all the p multipliers $\lambda_{r+1}, \lambda_{r+2}, \ldots, \lambda_m$ are nonviolators, a step along $P^r[-f_x]$ will violate at least one of the p constraints that are dropped. Hence it is necessary for the p constraints to contain at least one Kuhn-Tucker violator.

The result given by Theorem 1 is not surprising. Since λ_i is the rate of change of $f(x)$ with respect to g_i, when the status of the ith constraint is changed from active to inactive, the dropping of a constraint with negative multiplier would decrease $f(x)$. Also, if a number of constraints are being dropped, one would anticipate at least one of the constraints has a negative multiplier. It is, however, essential that the constraints being dropped must be safely dropped. The following lemma is an implication of the above theorem. It gives necessary and sufficient conditions for safely dropping a single constraint.

Lemma 1. Any constraint can be safely dropped singly if and only if the corresponding projection multiplier is negative.

Proof. Let the mth constant be dropped. A step along $P^{m-1}[-f_x]$ will violate (not violate) $g_m \leq 0$ iff (from eqn. (11) for $p = 1$)

$$[1 - G^T A_{m-1}^{-1} G] \lambda_m > 0 \ (< 0) \qquad (14)$$

Since

$$A_m = \begin{bmatrix} A_{m-1} & G \\ G^T & 1 \end{bmatrix}$$

the determinant of A_m in terms of its partitions is given by (Ref. 14)

$$\det (A_m) = (1 - G^T A_{m-1}^{-1} G) \det (A_{m-1})$$

Eqn. (14) becomes

$$\frac{\det (A_m)}{\det (A_{m-1})} \lambda_m > 0 \ (< 0) \tag{15}$$

In the normal case the gram determinants are positive (Ref. 15). So a step along $P^{m-1}[-f_x]$ violates (not violate) $g_m \leq 0$ if and only if $\lambda_m > 0$ (< 0).

Alternate proofs of this lemma are available in Refs. 3 and 9. The necessary and sufficient conditions of Lemma 1 have been extensively utilized in the literature on active-constraint logic. However, as mentioned earlier, more than one constraint often has to be dropped to satisfy the Kuhn-Tucker conditions. In such cases it is essential that the active-constraint logic used for determining the active set guarantees the nonviolation of the dropped constraints by the resulting step. Lemma 1 does not give this guarantee. The next theorem is a step towards resolving this problem. In a sequence of calculations, constraints satisfying certain conditions are dropped one at a time followed by recalculation of multipliers until either the Kuhn-Tucker conditions are met or no more constraints can be dropped.

Theorem 2. Constraints can be safely dropped one at a time followed by recalculation of multipliers provided the constraint being dropped satisfies the following:

(i) The projection multiplier of the constraint is negative.

(ii) In all the previous stages, where some other constraints were dropped, the multipliers corresponding to the constraint in question were all positive.

Proof. The proof is by induction. Let p be the number of constraints to be dropped.

$p = 1$. Application of lemma 1 trivially satisfies the theorem.

$p = 2$. Without loss of generality it is assumed that the m^{th} constraint was dropped first followed by the $m\text{-}1^{th}$ constraint. To apply the theorem, let

$$\lambda_m < 0$$

and

$$\lambda_{m-1} > 0 \quad , \quad \lambda_{m-1,m} < 0$$

where $\lambda_{m-1,m}$ is the projection multiplier corresponding to the $m\text{-}1^{th}$ constraint with the m^{th} constraint dropped.

Lemma 1 guarantees that the resulting step will not violate the $(m-1)^{th}$ constraint. So the first component of vector b in eqn. (12) is negative. Eqn. (13) yields

$$\lambda_{m-1}\, b_1 + \lambda_m\, b_2 > 0$$

This implies $b_2 < 0$, or the m^{th} constraint is also not violated.

Let the theorem be true for $p = k$.

$p = k+1$. Another constraint is dropped following the specified rules. Since the theorem is true for $p = k$, the last k constraints dropped will not be violated. So the first k components of the (k+1) dimensional vector b will be negative. Since

$$\lambda_i > 0 \quad , \quad i = k-1,\dots,m-1$$

and

$$\lambda_m < 0$$

eqn. (13) implies that the m^{th} constraint is also not violated.

It should be pointed out that the conditions of Theorem 2 are sufficient but not necessary for safely dropping constraints. Usually application of this theorem will not lead to an active-constraint set which satisfies Kuhn-Tucker condition, but it guarantees that none of the dropped constraints will be violated by the resulting step. Also, Theorem 2 does not determine uniquely an active set. At each stage a number of constraints may satisfy the conditions of the theorem and, depending on the constraints dropped, different active sets may be obtained. These active sets in general will have different number of constraints. In a special case (Corollary 1) this theorem unambiguously leads to the desired active set.

Corollary 1. If, at the initial and each subsequent stage of recalculation of multipliers, at most one of them has the wrong sign, then the constraint corresponding to the negative multiplier can be dropped without violating the constraints previously dropped.

The validity of Corollary 1 has been implicitly assumed by some authors. For this special case, Sargent's method of determining the active set (discussed in Chapter 1) holds.

With the help of Theorem 1 and Lemma 1, it is easy to generate a set of necessary and sufficient conditions for safely dropping a set of constraints. Out of the m constraints being satisfied as equalities, let the last p of them be dropped. Following the notation used in Theorem 1, consider the problem with the first r constraints and the j^{th} constraint taken as active, $j = r+1, r+2, \ldots, m$. Let $\tilde{\lambda}_j$ be the projection multiplier corresponding to the j^{th} constraint for this problem with r+1 active constraints. As before, the symbol $\hat{\lambda}_i$ will denote the i^{th} projection multiplier for the problem with first r constraints.

Lemma 2. The p constraints can be safely dropped and the projection multipliers for the resulting problem satisfy the Kuhn-Tucker conditions if and only if

$$\text{(i)} \quad \tilde{\lambda}_j < 0 \quad , \quad j = r+1, r+2, \ldots, m \tag{16}$$

$$\text{(ii)} \quad \hat{\lambda}_i > 0 \quad , \quad i = 1, 2, \ldots, r \tag{17}$$

Proof. Application of Lemma 1 to each of p subproblems, each with r+1 constraints, results in conditions (i). Conditions (ii) are the Kuhn-Tucker conditions for the problem with first r constraints.

In principle, Lemma 2 can be directly used to determine the desired active set. Constraints could be dropped first singly and then in sets of ones, twos, etc., each time checking the appropriate projection multipliers till the active set is found. This amounts to a brute-force method of obtaining the active set and would generally require a large computational effort. The major effort in the present work has been made in finding methods which would minimize the determination of intermediate projection multipliers and the associated testing.

The following lemma is a direct consequence of normality.

Lemma 3. $$\lambda^T \bar{\lambda} > 0 \tag{18}$$

Proof: ${g_x}^T g_x$ is the Grammian matrix of the vectors g_{1_x}, g_{2_x}, ..., g_{m_x}. For any nonzero m-vector y,

$$y^T ({g_x}^T g_x)\ y = (g_x\ y)^T\ (g_x\ y) \geq 0 \tag{19}$$

Assuming normality, the Gram determinant $\neq 0$; the Grammian is positive definite in the normal case.

From eqn. (3),

$$\bar{\lambda} = ({g_x}^T g_x)\ \lambda$$

or (20)

$$\lambda^T \bar{\lambda} = \lambda^T ({g_x}^T g_x)\ \lambda > 0\ .$$

The main implication of Lemma 3 is that at least one pair of multiplier components $(\lambda_i,\bar{\lambda}_i)$ must have the same sign. This fact will prove useful later in generating active constraint logic and in deciding on the existence and uniqueness of the desired active constraint set. The following lemma is a generalization of Lemma 3. Let $(A_{r+1})_j$ be the Gram matrix for the problem with the first r and the jth constraints $(j = r+1, r+2, \ldots, m)$. With A_r and $\tilde{\lambda}_i$ having the same meaning as before, the following holds.

Lemma 4. $$\sum_{j=r+1}^{m} \frac{\det((A_{r+1})_j)}{\det(A_r)} \tilde{\lambda}_j \lambda_j > 0 \tag{21}$$

Proof. Let the last p constraints be dropped and the remaining r constraints retained. From eqn. (15), the violation/nonviolation of the j^{th} constraint depends on the sign of

$$\frac{\det((A_{r+1})_j)}{\det(A_r)} \tilde{\lambda}_j \quad , \quad j = r+1,\ldots,m$$

Clearly the value of the above expression for $j = r+i$ is the same as the i^{th} component of vector b in eqn. (13), $i = 1,2,\ldots,p$. Making this substitution in eqn. (13) gives the desired result.

If all the constraints are dropped, $p = m$, $r = 0$ and $\tilde{\lambda} = \bar{\lambda}$, then

$$\det\,(A_0) = 1$$

and

$$\det\,(A_1)_j = g_{jj} = 1$$

Making these substitutions in eqn. (21) gives eqn. (20).

3.1 Conclusion

A number of general results have been presented in this chapter. These will prove useful in generating active-constraint logic in the following chapters.

4. THE TWO-CONSTRAINT CASE

4.1 Introduction

The special case of only two constraints being satisfied as equalities (m = 2) is discussed in this chapter. Many of the active-constraint logics found in literature work well here. The two most promising of them, the dual-violator rule and Sargent's worst-violator rule are discussed in the following two sections. An important question concerns the existence and uniqueness of an active constraint set with the desired properties. This question is treated in section 4.4, and the existence and uniqueness of the appropriate active set is established.

4.2 Dual-violator Rule

Let g_i and g_j be the two constraints satisfied as equalities. Using Lemma 1 and the implication of Lemma 3 that both the pairs $(\lambda_i,\bar{\lambda}_i)$ and $(\lambda_j,\bar{\lambda}_j)$ cannot have opposite signs, the active set can be obtained as follows:

(a) If $\lambda_i > 0$ and $\lambda_j > 0$, retain both constraints.

(b) For $\lambda_i > 0$ and $\lambda_j < 0$,

  (i) if $\bar{\lambda}_i > 0$ and $\bar{\lambda}_j > 0$, drop g_j

  (ii) if $\bar{\lambda}_i > 0$ and $\bar{\lambda}_j < 0$, drop g_j

  (iii) if $\bar{\lambda}_i < 0$ and $\bar{\lambda}_j < 0$, drop g_i and g_j

(c) For $\lambda_i < 0$ and $\lambda_j < 0$,

  (i) if $\bar{\lambda}_i > 0$ and $\bar{\lambda}_j < 0$, drop g_j

  (ii) if $\bar{\lambda}_i < 0$ and $\bar{\lambda}_j > 0$, drop g_i

(iii) if $\bar{\lambda}_i < 0$ and $\bar{\lambda}_j < 0$, drop g_i and g_j

Defining a dual violator as a constraint which has the wrong multiplier-component signs both singly and in combination with the other constraint, one can simplify the above logic to the following dual-violator rule:

I. If λ_i and $\lambda_j > 0$, retain both the constraints.

II. If $\bar{\lambda}_i$ and $\bar{\lambda}_j < 0$, drop both the constraints.

III. If a dual violator exists, drop it.

IV. If I, II and III do not apply and a violator exists, then drop it.

4.3 Sargent's Rule

In Sargent's method, the constraint corresponding to the most negative multiplier is dropped followed by recalculation of multipliers each time, until no negative multipliers remain. In its original form, this rule was ambiguous as the constraint gradients were not normalized (Ref. 9). Even when used along with normalized constraint gradients, it fails for m = 3. A counterexample to this effect is given in Appendix B. However, for m = 2 the modified Sargent's rule gives the desired active set. For two constraints, eqn. (20) yields

$$\left.\begin{aligned} \bar{\lambda}_i &= \lambda_i + (g_{i_x}{}^T g_{j_x})\, \lambda_j \\ \bar{\lambda}_j &= (g_{i_x}{}^T g_{j_x})\, \lambda_i + \lambda_j \end{aligned}\right\}$$

or

$$\bar{\lambda}_i - \bar{\lambda}_j = [1 - (g_{i_x}{}^T g_{j_x})]\, (\lambda_i - \lambda_j) \tag{22}$$

Since the constraint gradients have been normalized, it can be deduced from eqn. (22) that the same constraint has the smaller multiplier both when constraints are considered singly and together. Let the λ_i be the most negative component of λ. Then

$$\lambda_i < 0 \text{ and } \lambda_j > \lambda_i$$

Eqn. (22) gives

$$\bar{\lambda}_j > \bar{\lambda}_i$$

Using Sargent's rule, drop the i^{th} constraint. If $\bar{\lambda}_j$ is positive keep the j^{th} constraint. If $\bar{\lambda}_j$ is negative, then $\bar{\lambda}_i$ will necessarily be negative and both the constraints can be safely dropped.

4.4 Existence and Uniqueness of Desired Active Set

An implicit assumption in the proof of the dual-violator rule is the existence of the desired set of active constraints. The following theorem guarantees not only the existence but also the uniqueness of the active set.

Theorem 3: For a problem with m = 2, the desired active set given by the dual-violator rule always exists. Also, for a particular set of multipliers λ and $\bar{\lambda}$, this set is unique in the sense that only this active set satisfies the requirements of Lemma 2.

Proof. For the active set to be unique, only one of the following four cases can arise.

(i) $\lambda_i > 0$ and $\lambda_j > 0$ (active set: constraints i and j)

(ii) $\bar{\lambda}_i > 0$ and $\lambda_j < 0$ (active set: constraint i)

(iii) $\bar{\lambda}_j > 0$ and $\lambda_i < 0$ (active set: constraint j)

(iv) $\bar{\lambda}_i < 0$ and $\bar{\lambda}_j < 0$ (active set: no constraint)

Assume that (i) is true. Cases (ii) and (iii) are ruled out trivially. Using Lemma 3,

$$\lambda_i \bar{\lambda}_i + \lambda_j \bar{\lambda}_j > 0$$

and hence (iv) cannot occur. Similarly it can be shown that occurrence of (ii), (iii) or (iv) precludes the occurrence of other cases.

To prove the existence of an active set, it is sufficient to show that for each possible combination of signs of λ_i, λ_j, $\bar{\lambda}_i$ and $\bar{\lambda}_j$ one of the four cases (i) to (iv) holds. There are four possibilities for the signs of λ_i and λ_j:

(1) $\lambda_i > 0$ and $\lambda_j > 0$, case (i) satisfied.

(2) $\lambda_i > 0$ and $\lambda_j < 0$. The subpossibilities for the signs of $\bar{\lambda}_i$ and $\bar{\lambda}_j$ are

(a) $\bar{\lambda}_i > 0$, $\bar{\lambda}_j > 0$, case (ii) satisfied

(b) $\bar{\lambda}_i > 0$, $\bar{\lambda}_j < 0$, case (ii) satisfied

(c) $\bar{\lambda}_i < 0$, $\bar{\lambda}_j < 0$, case (iv) satisfied

(3) $\lambda_i < 0$ and $\lambda_j > 0$. Similar to (2) above.

(4) $\lambda_i < 0$ and $\lambda_j < 0$. Similar to (2) above.

4.5 Conclusion

The case discussed in this chapter is the simplest nontrivial case. The existence and uniqueness of the desired active set has been shown and strategies to obtain the active set discussed.

5. THE THREE-CONSTRAINT CASE

5.1 Introduction

The complexity of the problem increases significantly even when the number of constraints increases only from two to three. Additional sets of projection multipliers corresponding to the two-constraint subproblems obtained by dropping any one of the constraints have to be considered. The dual-violator rule is simple and elegant but unfortunately it does not hold for problems with more than two constraints. Similarly Sargent's rule also breaks down when more than two constraints are satisfied as equalities. Counterexamples to the dual-violator rule and Sargent's rule are given in Appendix B.

In the remainder of this section, two theorems are presented which utilize the special structure of the problem with m = 3. These, along with the material developed in Chapter 3, are used in the next three sections to generate an active-constraint logic.

Let the three constraints be

$$g_1(x) \leq 0, \; g_2(x) \leq 0 \text{ and } g_3(x) \leq 0 \tag{23}$$

Definition. $\lambda_{i,j}$ is the projection multiplier corresponding to the ith constraint for the two constraint subproblems obtained by dropping the jth constraint.

For the two constraints, g_i and g_k ($k \neq i,j$), eqn. (3) yields

$$\lambda_{i,j} = \frac{1}{(1 - g_{ik}^2)} \{\bar{\lambda}_i - g_{ik}\,\bar{\lambda}_k\} \tag{24}$$

$$\lambda_{k,j} = \frac{1}{(1 - g_{ik}^{2})} \{\bar{\lambda}_k - g_{ik} \bar{\lambda}_i\} \qquad (25)$$

where $$g_{ik} = (g_{i_x}^{T} g_{k_x})$$

Under assumptions of normality and normalization of constraint gradients, $|g_{ik}| < 1$.

The necessary and sufficient conditions for dropping one, two or three constraints can be obtained by specializing Lemma 2 to the present case. These are

(i) Constraint i can be dropped alone successfully and Kuhn-Tucker conditions satisfied if and only if

$$\left.\begin{array}{ll} & \lambda_i < 0 \\ \text{and} & \\ & \lambda_{j,i} > 0,\ \lambda_{k,i} > 0 \end{array}\right\} \qquad (26)$$

(ii) Constraints i and j can be successfully dropped and Kuhn-Tucker satisfied iff

$$\left.\begin{array}{ll} & \lambda_{i,j} < 0,\ \lambda_{j,i} < 0 \\ \text{and} & \\ & \bar{\lambda}_k > 0 \end{array}\right\} \qquad (27)$$

(iii) All the three constraints can be successfully dropped iff

$$\bar{\lambda}_1 < 0,\ \bar{\lambda}_2 < 0 \text{ and } \bar{\lambda}_3 < 0 \qquad (28)$$

Eqns. (26) - (28) will be extensively used in the remainder of this chapter.

The magnitude and signs of the components of $\bar{\lambda}$ have a major influence on the active-constraint set. This is amply borne out by the next two theorems for the special case under discussion.

Theorem 4. If it is known a priori that the desired active set contains exactly one constraint, then this constraint corresponds to the largest component of $\bar{\lambda}$.

Proof. Since all the three constraints are not to be dropped at least one component of $\bar{\lambda}$ must be positive. Let $\bar{\lambda}_1$ be the component of $\bar{\lambda}$ with the largest magnitude. There are two possibilities regarding the sign of $\bar{\lambda}_1$

(a) $\bar{\lambda}_1 > 0$

Therefore

$$\lambda_{1,2} = \frac{1}{(1 - g_{13}{}^2)} \{\bar{\lambda}_1 - g_{13}\, \bar{\lambda}_3\} > 0 \tag{29}$$

$$\lambda_{1,3} = \frac{1}{(1 - g_{12}{}^2)} \{\bar{\lambda}_1 - g_{12}\, \bar{\lambda}_2\} > 0 \tag{30}$$

because $\bar{\lambda}_1 > |\bar{\lambda}_2|$ and $\bar{\lambda}_1 > |\bar{\lambda}_3|$

So constraint 1 cannot be sucessfully dropped in combination with either one of the other constraints. The only possibility left is to drop constraints 2 and 3.

(b) $\bar{\lambda}_1 < 0$ and $|\bar{\lambda}_1| > |\bar{\lambda}_2|$, $|\bar{\lambda}_1| > |\bar{\lambda}_3|$

Consider the following two subcases

(i) Only one component of $\bar{\lambda}$ is negative.

So $\bar{\lambda}_2$ and $\bar{\lambda}_3$ are positive. Let $\bar{\lambda}_2 > \bar{\lambda}_3$. Therefore

$$\lambda_{2,1} = \frac{1}{(1 - g_{23}{}^2)} \{\bar{\lambda}_2 - g_{23}\,\bar{\lambda}_3\} > 0 \tag{31}$$

and constraints 1 and 2 cannot be successfully dropped. Since $\bar{\lambda}_1$ is negative, dropping constraints 2 and 3 together will not satisfy Kuhn-Tucker condition. This leaves the possibility of dropping constraints 1 and 3.

(ii) Two components of $\bar{\lambda}$ are negative.
In addition to $\bar{\lambda}_1$, let $\bar{\lambda}_2$ be negative. Since two constraints have to be dropped, the only way Kuhn-Tucker condition can be satisfied is by dropping constraints 1 and 2.

The following corollary extends Theorem 4 to the general case of m constraints. The proof follows directly from this theorem.

Corollary. In the general case if exactly m-1 constraints are to be dropped, then the constraint to keep corresponds to the largest component of $\bar{\lambda}$.

The following theorem assumes that not all the components of λ are positive.

Theorem 5. If the component of $\bar{\lambda}$ with the largest magnitude is negative, then the corresponding constraint cannot be in the active set.

Proof. Let

$$\bar{\lambda}_1 < 0 \text{ and } |\bar{\lambda}_1| > |\bar{\lambda}_2|,\ |\bar{\lambda}_1| > |\bar{\lambda}_3|$$

Then

$$\lambda_{1,2} = \frac{1}{(1 - g_{13}^2)} \{\bar{\lambda}_1 - g_{13}\,\bar{\lambda}_3\} < 0 \tag{32}$$

and

$$\lambda_{1,3} = \frac{1}{(1 - g_{12}^2)} \{\bar{\lambda}_1 - g_{12}\,\bar{\lambda}_2\} < 0 \tag{33}$$

There are three possibilities regarding the number of constraints in the active set.

(a) Active set consists of two constraints.

Since $\lambda_{1,2}$ and $\lambda_{1,3}$ are negative, dropping constraint 2 or 3 singly will not satisfy Kuhn-Tucker conditions. So constraint 1 has to be dropped.

(b) Active set contains a single constraint.

Applying theorem 2, the constraint corresponding to the largest component of $\bar{\lambda}$ is to be retained. Hence constraint 1 is necessarily dropped.

(c) Active set contains no constraints.

The theorem is trivially satisfied.

Unless noted otherwise it is assumed in the remaining sections of this chapter that the trivial cases, in which all the constraints are either kept or dropped, do not occur. The former corresponds to all the components of λ being positive while the latter corresponds to all components of $\bar{\lambda}$ being negative. The active-constraint logic is generated in the next two sections. The two cases, corresponding to the sign of the component of $\bar{\lambda}$ with largest magnitude, are considered separately.

5.2 Component of $\bar{\lambda}$ with Largest Magnitude Is Negative

Theorem 5 yields a simple rule for determining the active set. Let $\bar{\lambda}_i < 0$ be the component of $\bar{\lambda}$ with largest magnitude. Since all the three constraints are not to be dropped, at least one component of $\bar{\lambda}$ will be positive. Let the largest positive component of $\bar{\lambda}$ be $\bar{\lambda}_j$. Clearly, if only one constraint is to be dropped, it will be the i^{th} constraint, whereas if two constraints are to be dropped, they will be constraints i and k. The following rule results

(i) If $\lambda_{j,i} > 0$ and $\lambda_{k,i} > 0$, drop constraint i.

(ii) Else, drop constraints i and k.

5.3 Component of $\bar{\lambda}$ with Largest Magnitude Is Positive

Theorem 4 will prove useful here. It does not, however, lead directly to the active constraint logic as Theorem 5 did for the case discussed in the previous section.

For m = 3, eqn. (20) yields

$$\begin{bmatrix} \bar{\lambda}_1 \\ \bar{\lambda}_2 \\ \bar{\lambda}_3 \end{bmatrix} = \begin{bmatrix} 1 & g_{12} & g_{13} \\ g_{12} & 1 & g_{23} \\ g_{13} & g_{23} & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \tag{34}$$

Substituting for $\bar{\lambda}_i$ and $\bar{\lambda}_k$ in eqn. (24) from eqn. (34) repeatedly for various values of i, j and k from 1 to 3, $i \neq j \neq k$,

$$\lambda_{2,1} = \frac{1}{1 - g_{23}^2}((g_{12} - g_{13}\, g_{23})\,\lambda_1 + (1 - g_{23}^2)\,\lambda_2) \tag{35}$$

$$\lambda_{3,1} = \frac{1}{1 - g_{23}^2}((g_{13} - g_{12}\, g_{23})\,\lambda_1 + (1 - g_{23}^2)\,\lambda_3) \tag{36}$$

$$\lambda_{1,2} = \frac{1}{1 - g_{13}^2}((1 - g_{13}^2)\,\lambda_1 + (g_{12} - g_{13}\, g_{23})\,\lambda_2) \tag{37}$$

$$\lambda_{3,2} = \frac{1}{1 - g_{13}^2}((g_{23} - g_{12}\, g_{13})\,\lambda_2 + (1 - g_{13}^2)\,\lambda_3) \tag{38}$$

$$\lambda_{1,3} = \frac{1}{1 - g_{12}^2}((1 - g_{12}^2)\,\lambda_1 + (g_{13} - g_{12}\, g_{23})\,\lambda_3) \tag{39}$$

$$\lambda_{2,3} = \frac{1}{1 - g_{12}^2}((1 - g_{12}^2)\,\lambda_2 + (g_{23} - g_{12}\, g_{13})\,\lambda_3) \tag{40}$$

Three subpossibilities regarding the signs of the projection multipliers λ_1, λ_2 and λ_3 are considered.

(i) One projection multiplier is positive.

Let

$$\lambda_1 < 0,\ \lambda_2 < 0 \text{ and } \lambda_3 > 0$$

For the assumed signs of the projection multipliers, eqns. (35) - (40) result in

$$\lambda_{2,1} \gtrless 0 \Longrightarrow \frac{|\lambda_2|}{|\lambda_1|} \lessgtr - \frac{(g_{12} - g_{13}\, g_{23})}{(1 - g_{23}^2)} \tag{41}$$

$$\lambda_{3,1} \gtrless 0 \Longrightarrow \frac{\lambda_3}{|\lambda_1|} \gtrless \frac{(g_{13} - g_{12}\, g_{23})}{(1 - g_{23}^2)} \tag{42}$$

$$\lambda_{1,2} \gtrless 0 \Longrightarrow \frac{|\lambda_1|}{|\lambda_2|} \lessgtr - \frac{(g_{12} - g_{13}\, g_{23})}{(1 - g_{13}^{2})} \tag{43}$$

$$\lambda_{3,2} \gtrless 0 \Longrightarrow \frac{\lambda_3}{|\lambda_2|} \gtrless \frac{(g_{23} - g_{12}\, g_{13})}{(1 - g_{13}^{2})} \tag{44}$$

$$\lambda_{1,3} \gtrless 0 \Longrightarrow \frac{|\lambda_1|}{\lambda_3} \lessgtr \frac{(g_{13} - g_{12}\, g_{23})}{(1 - g_{12}^{2})} \tag{45}$$

$$\lambda_{2,3} \gtrless 0 \Longrightarrow \frac{|\lambda_2|}{\lambda_3} \lessgtr \frac{(g_{23} - g_{12}\, g_{13})}{(1 - g_{12}^{2})} \tag{46}$$

Let $\lambda_{2,1} > 0$. If $\lambda_{1,2} > 0$, then eqns. (41) and (43) yield

$$\frac{|\lambda_2|}{|\lambda_1|} \frac{|\lambda_1|}{|\lambda_2|} < \frac{(g_{12} - g_{13}\, g_{23})^2}{(1 - g_{23}^{2})\,(1 - g_{13}^{2})}$$

or

$$1 - g_{12}^{2} - g_{13}^{2} - g_{23}^{2} + 2\, g_{12}\, g_{13}\, g_{23} < 0 \tag{47}$$

The expression on the left hand side in eqn. (47) is the gram determinant of g_x for the three constraint problems and hence cannot be negative.

Therefore

$$\lambda_{2,1} > 0 \Longrightarrow \lambda_{1,2} < 0 \tag{48}$$

In a similar fashion it can be shown, using eqns. (41) to (46), that the following is true.

$$\lambda_{3,1} < 0 \Longrightarrow \lambda_{1,3} < 0 \tag{49}$$

$$\lambda_{1,2} > 0 \Longrightarrow \lambda_{2,1} < 0 \tag{50}$$

$$\lambda_{3,2} < 0 \Longrightarrow \lambda_{2,3} < 0 \tag{51}$$

$$\lambda_{1,3} > 0 \Longrightarrow \lambda_{3,1} > 0 \tag{52}$$

$$\lambda_{2,3} > 0 \Longrightarrow \lambda_{3,2} > 0 \tag{53}$$

Any one of the three components of $\bar{\lambda}$ could have the largest magnitude (it has been assumed to be positive). These three cases are considered separately.

(a) Let $\bar{\lambda}_1 > 0$ and $\bar{\lambda}_1 > |\bar{\lambda}_2|$, $\bar{\lambda}_1 > |\bar{\lambda}_3|$

Then

$$\lambda_{1,2} = \frac{1}{(1 - g_{13}^2)} \{\bar{\lambda}_1 - g_{13}\,\bar{\lambda}_3\} > 0 \tag{54}$$

$$\lambda_{1,3} = \frac{1}{(1 - g_{12}^2)} \{\bar{\lambda}_1 - g_{12}\,\bar{\lambda}_2\} > 0 \tag{55}$$

Eqns. (50) and (52) give

$$\lambda_{2,1} < 0 \text{ and } \lambda_{3,1} > 0$$

If a single constraint has to be dropped, it cannot be constraint 1 because $\lambda_{2,1} < 0$. It cannot be constraint 3 either as $\lambda_3 > 0$. So, by elimination, it must be constraint 2. If two constraints are to be dropped, they must be constraints 2 and 3 (Theorem 4). Keeping in mind that $\lambda_{1,2} > 0$, the following rule results

If $\lambda_{3,2} > 0$ drop constraint 2.

Else drop constraints 2 and 3.

(b) Let $\bar{\lambda}_2 > 0$ and $\bar{\lambda}_2 > |\bar{\lambda}_1|$, $\bar{\lambda}_2 > |\bar{\lambda}_3|$

This is similar to case (a) and the following rule results

If $\lambda_{3,1} > 0$ drop constraint 1.

Else drop constraints 1 and 3.

(c) Let $\bar{\lambda}_3 > 0$ and $\bar{\lambda}_3 > |\bar{\lambda}_1|$, $\bar{\lambda}_3 > |\bar{\lambda}_2|$

Then

$$\lambda_{3,1} = \frac{1}{(1 - g_{23}{}^2)} \{\bar{\lambda}_3 - g_{23}\,\bar{\lambda}_2\} > 0 \tag{56}$$

$$\lambda_{3,2} = \frac{1}{(1 - g_{13}{}^2)} \{\bar{\lambda}_3 - g_{13}\,\bar{\lambda}_1\} > 0 \tag{57}$$

So neither constraints 1 and 3 nor constraints 2 and 3 can be dropped together. This is also implied by Theorem 4. Keeping in mind that both $\lambda_{1,2}$ and $\lambda_{2,1}$ cannot be positive together (eqns. (48) and (50)), the following rule can be used to obtain the active set.

If $\lambda_{1,2} > 0$, drop g_2 alone.

If $\lambda_{2,1} > 0$, drop g_1 alone.

Else drop both g_1 and g_2.

(ii) Two projection multipliers are positive

Let

$$\lambda_1 > 0,\ \lambda_2 > 0 \text{ and } \lambda_3 < 0$$

In a manner and analogous to that used in (i), it can be shown that

$$\lambda_{2,1} < 0 \implies \lambda_{1,2} > 0 \tag{58}$$

$$\lambda_{3,1} > 0 \implies \lambda_{1,3} > 0 \tag{59}$$

$$\lambda_{1,2} < 0 \implies \lambda_{2,1} > 0 \tag{60}$$

$$\lambda_{3,2} > 0 \implies \lambda_{2,3} > 0 \tag{61}$$

$$\lambda_{1,3} < 0 \implies \lambda_{3,1} < 0 \tag{62}$$

$$\lambda_{2,3} < 0 \Longrightarrow \lambda_{3,2} < 0 \tag{63}$$

As in (i), the following three cases are considered

(a) Let $\bar{\lambda}_1 > 0$ and $\bar{\lambda}_1 > |\bar{\lambda}_2|$, $\bar{\lambda}_1 > |\bar{\lambda}_3|$

Then from eqns. (54) and (55)

$$\lambda_{1,2} > 0,\ \lambda_{1,3} > 0 \tag{64}$$

Since λ_1 and λ_2 are positive, both constraints 1 and 2 cannot be dropped singly. If two constraints have to be dropped, they must be constraints 2 and 3 (Theorem 4). So the following rule can be used

If $\lambda_{2,3} > 0$, drop g_3 alone.

Else drop g_2 and g_3.

Note. Since $\lambda_{2,3} < 0$ implies $\lambda_{3,2} < 0$, both constraints 2 and 3 can be safely dropped together.

(b) Let $\bar{\lambda}_2 > 0$ and $\bar{\lambda}_2 > |\bar{\lambda}_1|$, $\bar{\lambda}_2 > |\bar{\lambda}_3|$

This case is similar to (a).

(c) Let $\bar{\lambda}_3 > 0$ and $\bar{\lambda}_3 > |\bar{\lambda}_1|$, $\bar{\lambda}_3 > |\bar{\lambda}_2|$

Eqns. (56) and (57) give

$$\lambda_{3,1} > 0,\ \lambda_{3,2} > 0 \tag{65}$$

Eqn. (65) along with eqns. (59) and (61) give

$$\lambda_{1,3} > 0 \text{ and } \lambda_{2,3} > 0 \tag{66}$$

Since λ_3 is negative, eqn. (66) implies that the active set is obtained by dropping constraint 3.

(iii) All projection multipliers are negative.

It can be shown (as in (i)) that

$$\lambda_{2,1} > 0 \Longrightarrow \lambda_{1,2} < 0 \tag{67}$$

$$\lambda_{3,1} > 0 \Longrightarrow \lambda_{1,3} < 0 \tag{68}$$

$$\lambda_{1,2} > 0 \Longrightarrow \lambda_{2,1} < 0 \tag{69}$$

$$\lambda_{3,2} > 0 \Longrightarrow \lambda_{2,3} < 0 \tag{70}$$

$$\lambda_{1,3} > 0 \Longrightarrow \lambda_{3,1} < 0 \tag{71}$$

$$\lambda_{2,3} > 0 \Longrightarrow \lambda_{3,2} < 0 \tag{72}$$

Let $\bar{\lambda}_1 > 0$ and $\bar{\lambda}_1 > |\bar{\lambda}_2|$, $\bar{\lambda}_1 > |\bar{\lambda}_3|$

Eqns. (54) and (55) hold, which along with eqns. (69) and (71) give

$$\lambda_{1,2} > 0,\ \lambda_{2,1} < 0,\ \lambda_{1,3} > 0 \text{ and } \lambda_{3,1} < 0$$

Since $\lambda_{2,1}$ and $\lambda_{3,1}$ are negative, constraint 1 cannot be dropped singly. If two constraints have to be dropped, they must be constraints 2 and 3. Keeping in mind that both $\lambda_{3,2}$ and $\lambda_{2,3}$ cannot be positive together, the following rule can be used.

If $\lambda_{3,2} > 0$ drop g_2.
If $\lambda_{2,3} > 0$ drop g_3.
Else drop g_2 and g_3.

The cases corresponding to $\bar{\lambda}_2$ or $\bar{\lambda}_3$ having the largest magnitude are similar to the one discussed.

5.4 Active-constraint Logic

The results presented in sections 5.2 and 5.3 can be combined to give the following active constraint logic for m = 3.

The following steps have to be examined sequentially, the first one that cannot be eliminated gives the desired active set.

I. If λ_i, λ_j, $\lambda_k > 0$, keep all the constraints.

II. If $\bar{\lambda}_i$, $\bar{\lambda}_j$, $\bar{\lambda}_k < 0$, drop all the constraints.

III. Let the largest and smallest components of $\bar{\lambda}$ be $\bar{\lambda}_j$ and $\bar{\lambda}_i$ respectively.

(a) For $|\bar{\lambda}_i| > \bar{\lambda}_j$ (else go to (b))

(i) if $\lambda_{j,i} > 0$ and $\lambda_{k,i} > 0$ drop constraint i

(ii) else drop constraints i and k

(b) (i) if $\lambda_i > 0$ and $\lambda_k > 0$ drop constraint j

(ii) if $\lambda_i < 0$ and $\lambda_{k,i} > 0$ drop constraint i

(iii) if $\lambda_k < 0$ and $\lambda_{i,k} > 0$ drop constraint k

(iv) else drop constraints i and k

This logic requires the computation of at most two intermediate projection multipliers to determine the set of active constraints.

5.5 Existence and Uniqueness of Desired Active Set

Throughout the previous sections of this chapter, it has been tacitly assumed that the desired active set always exists.

Specializing Lemma 4 for m = 3 and p = 2, the following results are obtained

$$\text{(i)} \quad (1 - g_{13}^2)\,\lambda_1\,\lambda_{1,2} + (1 - g_{23}^2)\,\lambda_2\,\lambda_{2,1} > 0 \qquad (73)$$

$$\text{(ii)} \quad (1 - g_{12}^2)\,\lambda_1\,\lambda_{1,3} + (1 - g_{23}^2)\,\lambda_3\,\lambda_{3,1} > 0 \qquad (74)$$

$$\text{(iii)} \quad (1 - g_{12}^2)\,\lambda_2\,\lambda_{2,3} + (1 - g_{13}^2)\,\lambda_3\,\lambda_{3,2} > 0 \qquad (75)$$

Applying Lemma 3 to the three-constraint problem and the three two-constraint subproblems obtained by dropping any one of the constraints results in the following

(i) $\lambda_1 \bar{\lambda}_1 + \lambda_2 \bar{\lambda}_2 + \lambda_3 \bar{\lambda}_3 > 0$ (76)

(ii) $\lambda_{1,3} \bar{\lambda}_1 + \lambda_{2,3} \bar{\lambda}_2 > 0$ (77)

(iii) $\lambda_{1,2} \bar{\lambda}_1 + \lambda_{3,2} \bar{\lambda}_3 > 0$ (78)

(iv) $\lambda_{2,1} \bar{\lambda}_2 + \lambda_{3,1} \bar{\lambda}_3 > 0$ (79)

Since the constraints have been normalized, eqns. (73) - (75) imply that both the pairs $(\lambda_i, \lambda_{i,j})$ and $(\lambda_j, \lambda_{j,i})$ cannot have opposite signs. Similarly eqns. (77) to (79) imply that both the pairs $(\lambda_{i,k}, \lambda_i)$ and $(\lambda_{j,k}, \bar{\lambda}_j)$ cannot have opposite signs.

For any given set of signs for λ_1, λ_2, λ_3, $\bar{\lambda}_1$, $\bar{\lambda}_2$, $\bar{\lambda}_3$ compatible with eqn. (76), there are 64 possible cases corresponding to the various signs taken by the six intermediate projection multipliers $\lambda_{i,j}$ $(i,j = 1,\ldots,3,\ i \neq j)$. Out of these 64 possible cases, 37 cases are ruled out by eqns. (77) - (79). Depending on the signs of λ_i, $\bar{\lambda}_i$, $i = 1,\ldots,3$, some more cases will be ruled out. It is the contention that each of the remaining cases corresponds to some active set which satisfies the conditions of Lemma 2. Also each remaining case yields exactly one desired active set. This has been shown to be true for two different sets of signs of the components of λ and $\bar{\lambda}$ in Appendix 3. Thus, for a given λ and $\bar{\lambda}$, it can be shown that the active set always exists and each possible case corresponds to a (can be different) single active set.

5.6 Conclusion

The complexity of the problem of determining the desired active set even when only three constraints are satisfied as equalities is evident. The active-constraint logic obtained always leads to a unique active set.

6. THE GENERAL CASE

In a general nonlinear programming problem, any number of constraints, from zero to n (number of variables in the problem), can be satisfied as equalities at a given point. The rule suggested by Theorem 2 can be used to obtain an active set of constraints. This active set will, in general, possess only one of the two attributes of the desired active set mentioned in Lemma 2. Although one or more constraints can be dropped with the guarantee that the resulting step will not violate the dropped constraints, there is no matching guarantee that the active set obtained will satisfy Kuhn-Tucker conditions.

There is a certain amount of freedom regarding the set of constraints which can be dropped using Theorem 2. Constraints which satisfy the conditions of Theorem 2 are dropped one at a time followed by recalculation of multipliers until no more constraints can be safely dropped. Generally, more than one constraint will satisfy the conditions of the theorem, and depending on which of these constraints is dropped different active sets will result. In an attempt to find an active set which satisfies the Kuhn-Tucker conditions or an active set with the least number of constraints, the various active sets could be found and checked. This is not recommended since not only would it require much computing but there is the possibility that none of these active sets satisfy the Kuhn-Tucker conditions. It seems appropriate to drop the worst violator which satisfies the conditions of Theorem 2.

The following active constraint logic is suggested.

I. If all the multipliers have the 'right' sign, keep all the constraints.

II. If all the components of $\bar{\lambda}$ have the 'wrong' sign, drop all the constraints.

III. If $m \leq 3$, use the logic developed in Chapters 4 and 5.

IV. For $m > 3$, use Theorem 2 repeatedly, each time dropping the worst violator which satisfies its conditions.

Since it is guaranteed that the resulting step will not violate any of the constraints that were dropped, it is expected that this active set strategy will not cause cycling of constraints between active and inactive status.

7. COMPUTATIONAL STUDY

7.1 Introduction

The computational study was done to assess the effectiveness of existing active set strategies in comparison to the one proposed in this study. Only quadratic programming problems have been considered.

The optimization algorithm used is essentially the quadratic programming method suggested by Powell (Ref. 3). The problem is

Minimize $$f(x) = \frac{1}{2} x^T Q x + q^T x + C \tag{80}$$

subject to linear inequalities

$$A^T x \geq d \tag{81}$$

where

Q is an n x n positive definite symmetric matrix

A is an n x M matrix

q is an n-vector

d is an M-vector

and c is a constant

At any feasible point x, m ($\leq$ n) of the constraints are satisfied as equalities. It is assumed that these m constraints are linearly independent. The minimum of f(x) subject to these m equality constraints is given by f(x + s) where the 'step' s is

$$s = \{Q^{-1} \hat{A} (\hat{A}^T Q^{-1} \hat{A})^{-1} \hat{A}^T Q^{-1} - Q^{-1}\} f_x \tag{82}$$

where $f_x = Qx + q$

and $\hat{A}$ is an n x m matrix containing those columns of A which correspond to the m constraints considered.

The Lagrange multipliers at x are given by

$$\lambda = (\hat{A}^T Q^{-1} \hat{A})^{-1} \bar{\lambda} \tag{83}$$

where $\bar{\lambda} = \hat{A}^T Q^{-1} f_x$

The basic steps of the algorithm used for minimizing a strictly convex quadratic function subject to linear inequalities are given below. It is assumed that the initial point is feasible.

(a) Determine the set of constraints satisfied as equalities at the initial point.

(b) Compute the multipliers and determine the set of active constraints.

(c) Compute the step given by vector s. If the length of this step is less than a preassigned value ϵ, go to step (e).

(d) If the point x + s is feasible, determine the multipliers at this point and go to step (e). Else, calculate the largest value of α such that $x + \alpha s$ is feasible. Add the new constraints which are satisfied as equalities to the active set. Go to (b).

(e) If all multipliers are positive, stop. Else determine the set of active constraints and go to (c).

Appendix D contains a listing of the computer program.

7.2 Active-set Strategies

The proposed active-set strategy is compared against three existing strategies. These strategies were chosen so as to cover the complete spectrum of active-set strategies from the least restrictive to the most restrictive. Any active constraint logic has to decide on adding and dropping constraints. In all the active constraint logics compared, a new constraint is added to the active set as soon as it is satisfied as an equality. The difference between the various strategies exists in dropping constraints from the active set. The proposed active-constraint logic is discussed in Chapter 6. The three other strategies employed in the comparison are described below.

I. Sargent's worst-violator strategy.

In this strategy, every time a constraint is added or the minimum with the constraints in the active set taken as equalities reached, the active set is determined anew. The worst violator (constraint with the most negative multiplier) is dropped followed by recalculation of the multipliers. This is repeated till no violators remain in the active set. This logic is used in combination with Zoutendijk's anticycling rule (see Chapter 1).

II. Fletcher's active-set strategy.

In Fletcher's approach, constraints are dropped only when absolutely necessary for continued progress. If the Kuhn-Tucker conditions are not met at the minimum of the objective function with the constraints in the active set satisfied as equalities, then the worst violator is dropped singly.

III. Rosen's active-set strategy.

This strategy represents a compromise between the two extremes represented by I and II. Also the constraints are dropped on the basis of additional decrease in objective function obtained by dropping them rather than on the size of the multipliers. Whenever a constraint is added to the active set, an estimate is made of the additional decrease inthe objective function obtained by dropping violators singly. Let $x^{(1)}$ be the current point and assume that m constraints are satisfied as equalities. Assume that the minimum of f(x) with these m constraints treated as equalities occurs at $x^{(2)}$ and that the minimum with m-1 constraints treated as equalities (a single violator is dropped, say the i^th^ constraint) is at $x^{(3)}$. Let the value of the objective function at $x^{(k)}$ be F_k. Clearly $F_1 - F_2$ is the decrease in the objective function by keeping all the constraints satisfied as equalities in the active set and $F_2 - F_3$ is the additional decrease obtained by dropping the ith constraint (assumed to be a violator). Then, for quadratic programming (Ref. 4),

$$F_1 - F_2 = \frac{1}{2} s^T G s \tag{84}$$

and

$$F_2 - F_3 = \frac{1}{2} \lambda_i^2/u_{ii} \tag{85}$$

where $s = x^{(2)} - x^{(1)}$

and u_{ii} is the ith diagonal element of the matrix $(\hat{A} G^{-1} \hat{A})^{-1}$. From among all the Kuhn-Tucker violators, the constraint which gives the maximum $F_2 - F_3$ is deleted from the active set if

$$(F_2 - F_3)_{max} \geq (F_1 - F_2) \; . \tag{86}$$

This ensures that $F_1 - F_3 \geq 2(F_1 - F_2)$.

7.3 Results

The test problems used were quadratic programming problems taken mainly from Ref. 16. Each test problem was solved using three different initial points and each of these twelve cases were solved using the active-set logics of Fletcher, Sargent, Rosen and the one proposed in this study. These logics were compared for the effect they had on the number of times the gradient of the objective function was evaluated.

The functions f, g_1, g_2, ..., g_m used along with the results of the computational study are given in Tables 1-4. As expected, the most-constrained logic of Fletcher generally required the most gradient evaluations. Rosen's rule always performed as well or better than Fletcher's rule. The two least-constrained logics (Sargent's worst violator rule and the one proposed here) performed identically if $m \leq 2$ at all points encountered during the constrained minimization. This is not surprising, as Sargent's rule, when used with normalized constraint gradients, gives the desired active set for $m \leq 2$. When $m \geq 3$, there is the possibility of the "wrong" active set being chosen by Sargent's rule and this in turn could lead to a much larger number of f_x evaluations. This is borne out in cases 8 and 11. In case 12, Sargent's rule by chance obtains the desired active set resulting in a lower number of f_x evaluations as compared to the other active-constraint logics.

8. CONCLUSIONS

The problem of selecting the set of active constraints from among the constraints satisfied as equalities has been discussed in detail. Since the least-constrained strategies have generally proven to be superior to the most-constrained strategies, the major effort in this study has been in generating an active-constraint strategy which does not have the deficiencies present in the available least-constrained strategies. The problem boils down to selecting a subset of the set of constraints satisfied as equalities such that all the constraints in this active-set satisfy the Kuhn-Tucker conditions and the resulting step does not violate any of the dropped constraints.

It has been shown that this problem is relatively simple when only two constraints are satisfied as equalities ($m = 2$). Some of the existing active-constraint logics work well for this special case. The problem becomes much more complex when m increases even from two to only three. With the help of some of the general results developed here and the special structure of the problem for $m = 3$, an active-set strategy has been given for this case. This strategy is not as simple as one would like. It was also found that for both $m = 2$ and 3, the desired active set at a given point is unique. The existence of this set is also guaranteed.

For the problem with more than three constraints satisfied as equalities, the rule suggested in this study does not in general give an active-set which satisfies the Kuhn-Tucker conditions. However, it guarantees that none of the dropped constraints is violated by the resulting step and hence prevents cycling of constraints between active and inactive status. A limited amount of testing with quadratic-programming problems showed that the proposed active-set strategy almost always equaled or outperformed the best among the three existing strategies tested against it.

REFERENCES

1. Lenard, M. L., "A Computational Study of Active-Set Strategies in Nonlinear Programming with Linear Constraints," Mathematical Programming 16, 1979, pp. 81-97.

2. Fletcher, R., A General Quadratic Programming Algorithm," Journal of the Institute of Mathematics and Applications, 1971, pp. 76-91.

3. Powell, M. J. D., "Introduction to Constrained Optimization," in Numerical Methods for Constrained Optimization, Gill, P. E., and Murray, W., editors, Academic Press, New York, 1974, pp. 1-28.

4. Gill, P. E. and Murray, W., "Quasi-Newton Methods for Linearly Constrained Optimization," in Numerical Methods for Constrained Optimization, Gill, P. E., and Murray, W., editors, Academic Press, New York, 1974, pp. 67-90.

5. Gill, P. E. and Murray, W., "The Computation of Lagrange Multiplier Estimates for Constrained Minimization," Mathematical Programming, Vol. 17, 1979, pp. 32-60.

6. Kelley, H. J., Lefton, L., and Johnson, I. L., "Curvilinear-Projection Developments," Journal of Guidance and Control, January 1978.

7. Kelley, H. J., and Lefton, L., "Variable-Metric Projection Optimization Program Notes," Analytical Mechanics Associates, Inc. Report, April 1978.

8. Sargent, R. W. H., and Murtagh, B. A., "Projection Methods for Non-Linear Programming," Mathematical Programming 4, 1973, pp. 245-268.

9. Sargent, R. W. H., "Reduced-gradient and Projection Methods for Nonlinear Programming," in Numerical Methods for Constrained Optimization, Gill, P. E., and Murray, W., eds., Academic Press, New York, 1974, pp. 149-174.

10. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming. Part I: Linear Constraints," SIAM Journal, Vol. 8, No. 1, 1960, pp. 181-217.

11. Zoutendijk, G., Methods of Feasible Directions, Elsevier, Amsterdam, 1960.

12. Gill, P. E., and Murray, W., "Newton-type Methods for Linearly Constrained Optimization," in Numerical Methods for Constrained Optimization, Gill, P. E., and Murray, W., editors, Academic Press, New York, 1974, pp. 29-66.

13. McCormick, G. P., "A Second Order Method for the Linearly Constrained Nonlinear Programming Problem," in Nonlinear Programming, Rosen, J. B., Mangasarian, O. L., and Ritter, K., editors, Academic Press, New York, pp. 207-243.

14. Brogan, W. L., Modern Control Theory, Quantum Publishers, Inc., 1974, pp. 48-55.

15. Luenberger, D. G., Optimization by Vector Space Methods, Wiley, 1969, pp. 56-57.

16. Hock, W. and Schittkowski, K., Test Examples for Nonlinear Programming Codes, Springer-Verlag, 1981.

17. Kuhn, H. W. and Tucker, A. W., "Nonlinear Programming," in Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, Neyman, J., editor, University of California Press, Berkeley, 1951, pp. 481-492.

18. Karush, W., "Minima of Functions of Several Variables with Inequalities as Side Conditions," Master's Thesis, Department of Mathematics, University of Chicago, 1939.

19. John, F., "Extremum Problems with Inequalities as Subsidiary Conditions," in Studies and Essays, Courant Anniversary Volume, Interscience, New York, 1951, pp. 187-204.

20. Takayama, A., Mathematical Economics, The Dryden Press, Hinsdale, Illinois, 1974, pp. 55-168.

21. Kuhn, H. W., "Nonlinear Programming: A Historical View," in Nonlinear Programming, SIAM-AMS Proceedings, Vol. IX, Cottle, R. W., and Lemke, C. E., editors, American Mathematical Society, Rhode Island, 1976, pp. 1-26.

22. Aoki, M., Introduction to Optimization Techniques, McMillan, New York, 1971 , pp. 181-189.

APPENDIX A

A function f(x) is to be minimized subject to M inequality constraints,

$$g_i(x) \leq 0 \quad , \quad i = 1,2,\ldots,M \tag{A1}$$

m of which are satisfied as equalities at the current point x (n-vector). x is assumed to be a feasible point. In the usual gradient-projection method, a linear approximation to f(x) is minimized subject to linearized constraints and a quadratic constraint on step size

$$\text{Minimize} \qquad f(\bar{x}) + f_x^T \, \Delta x \tag{A2}$$

Subject to

$$g_{i_x}^T \, \Delta x = 0 \quad , \quad i = 1,2,\ldots,m \tag{A3}$$

and

$$\Delta x^T \, H \, \Delta x - s^2 = 0 \tag{A4}$$

where $\bar{x}$ is a feasible point and H is a positive definite symmetric matrix.

The Lagrangian is given by

$$L = f(\bar{x}) + f_x^T \, \Delta x + \sum_{i=1}^{m} \lambda_i \, g_{i_x}^T \, \Delta x + \lambda_0 (\Delta x^T \, H \, \Delta x - s^2) \tag{A5}$$

Putting $\frac{\partial L}{\partial \Delta x} = 0$ yields

$$H \, \Delta x = - \frac{1}{2\lambda_0} (f_x + g_x \, \lambda) \text{ for } \lambda_0 \neq 0 \tag{A6}$$

or

$$\Delta x = - \frac{H^{-1}}{2\lambda_0} (f_x + g_x \, \lambda) \tag{A7}$$

Substituting Δx from eqn. A7 into

$$g_x^T \Delta x = 0 \qquad \text{A8}$$

gives

$$g_x^T H^{-1} f_x + g_x^T H^{-1} g_x \lambda = 0 \qquad \text{A9}$$

or

$$\lambda = - (g_x^T H^{-1} g_x)^{-1} g_x^T H^{-1} f_x \qquad \text{A10}$$

The matrix $g_x^T H^{-1} g_x$ will be nonsingular if the m constraint gradients g_{i_x}, i = 1,2,...,m are linearly independent.

APPENDIX B

Counterexample to Sargent's Rule.

Consider the following example with n = 4 and m = 3. At the point under consideration, the gradients of the constraints and the objective function are given by the following

$$g_{1_x} = [\ 1 \quad 0 \quad 0 \quad 0\ ]^T \tag{B1}$$

$$g_{2_x} = [.375 \quad .1199 \quad .65 \quad .65\ ]^T \tag{B2}$$

$$g_{3_x} = [.875 \quad 0 \quad .4335 \quad .2155]^T \tag{B3}$$

$$f_x = [\ -1 \quad 8.8624 \quad 2.8836 \quad 0\ ]^T \tag{B4}$$

The constraint gradients have been normalized and are linearly independent. Using equations (3) and (4), the various projection multipliers corresponding to problems with 1, 2 and 3 constraints can be calculated.

$$\begin{array}{llllll}
\bar{\lambda}_1 & = 1.0000, & \bar{\lambda}_2 & = -.8750, & \bar{\lambda}_3 & = .7500 \\
\lambda_{2,1} & = -3.2857, & \lambda_{3,1} & = 3.2143, & \lambda_{1,2} & = 1.4667 \\
\lambda_{3,2} & = -.5333, & \lambda_{1,3} & = 1.5455, & \lambda_{2,3} & = -1.4545 \\
\lambda_1 & = -10.8334, & \lambda_2 & = -10.2500, & \lambda_3 & = 17.9167
\end{array}$$

Using Sargent's rule of dropping the constraint with the most negative multiplier, constraint 1 is dropped first, followed by constraint 2. Since $\bar{\lambda}_3$ is positive, Sargent's rule gives the active constraint set consisting of constraint 3. This is not the desired active set because a step along the projected gradient will violate constraint 1. The desired active set is obtained by applying the logic developed in Chapter 4 and consists of constraint 1.

Counterexample to Dual-Violator Rule.

The following example is also with n = 4 and m = 3. The constraints and objective function gradients at the point under consideration are

$$\begin{aligned}
g_{1x} &= [\ 1 \quad 0 \quad 0 \quad 0\ ]^T \\
g_{2x} &= [-.8200 \quad .1970 \quad -.3800 \quad -.3800]^T \\
g_{3x} &= [\ .4500 \quad 0 \quad .8093 \quad .3775]^T \\
f_x &= [\ 1 \quad .4358 \quad .3707 \quad 0\ ]^T
\end{aligned}$$

The constraints are linearly independent and are normalized. The projection multipliers corresponding to problems with 1, 2 and 3 constraints as active are

$$\begin{array}{lll}
\bar{\lambda}_1 = 1.0000 , & \bar{\lambda}_2 = -.8000 , & \bar{\lambda}_3 = .2000 \\
\lambda_{2,1} = -1.9414 , & \lambda_{3,1} = -1.3919 , & \lambda_{1,2} = 1.1411 \\
\lambda_{3,2} = -.3135 , & \lambda_{1,3} = 1.0501 , & \lambda_{2,3} = .0611 \\
\lambda_1 = .1955 , & \lambda_2 = -1.6782 , & \lambda_3 = -1.2630
\end{array}$$

Using the dual-violator rule, constraint 2 is first dropped, as it is the only dual violator. Recalculation of projection multipliers reveals that there is no dual violator but constraint 3 is a violator and hence it is dropped. Since $\bar{\lambda}_1$ is positive, the dual-violator rule gives the active constraint set consisting of constraint 1. This is not the desired set because the resulting step will violate constraint 2. The desired active constraint set consists of constraints 1 and 2.

APPENDIX C

The existence of the active constraint set for three-constraint problems is discussed here. With the help of two separate choices of the signs of λ and $\bar{\lambda}$, it is shown that all the 64 cases corresponding to the positive or negative sign taken by the 6 intermediate projection multipliers $\lambda_{i,j}$ are either ruled out by eqns. (73) - (79) or lead to an active set of constraints.

I. $\lambda_1 > 0$, $\lambda_2 > 0$, $\lambda_3 < 0$, $\bar{\lambda}_1 > 0$, $\bar{\lambda}_2 > 0$ and $\bar{\lambda}_3 > 0$ (C1)

The only possible active sets for a problem with the above set of signs for λ and $\bar{\lambda}$ are (from Lemma 2)

(i) Constraints 1 and 2.

(ii) If $\lambda_{2,3} < 0$ and $\lambda_{3,2} < 0$, then constraint 1.

(iii) If $\lambda_{1,3} < 0$ and $\lambda_{3,1} < 0$, then constraint 2.

In the following table, all possible signs of $\lambda_{i,j}$ are considered. The numbers in brackets in the comments column indicate the constraints in the active set.

Case No.	$\lambda_{2,1}$	$\lambda_{3,1}$	$\lambda_{1,2}$	$\lambda_{3,2}$	$\lambda_{1,3}$	$\lambda_{2,3}$	Comments
1	+	+	+	+	+	+	Active set (1,2)
2	+	+	+	+	+	-	Ruled out
3	+	+	+	+	-	+	Ruled out
4	+	+	+	+	-	-	Ruled out
5	+	+	+	-	+	+	Active set (1,2)
6	+	+	+	-	+	-	Active set (1)
7	+	+	+	-	-	+	Ruled out
8	+	+	+	-	-	-	Ruled out
9	+	+	-	+	+	+	Active set (1,2)
10	+	+	-	+	+	-	Ruled out

Case No.	$\lambda_{2,1}$	$\lambda_{3,1}$	$\lambda_{1,2}$	$\lambda_{3,2}$	$\lambda_{1,3}$	$\lambda_{2,3}$	Comments
11	+	+	-	+	-	+	Ruled out
12	+	+	-	+	-	-	Ruled out
13	+	+	-	-	+	+	Ruled out
14	+	+	-	-	+	-	Ruled out
15	+	+	-	-	-	+	Ruled out
16	+	+	-	-	-	-	Ruled out
17	+	-	+	+	+	+	Active set (1,2)
18	+	-	+	+	+	-	Ruled out
19	+	-	+	+	-	+	Active Set (2)
20	+	-	+	+	-	-	Ruled out
21	+	-	+	-	+	+	Active set (1,2)
22	+	-	+	-	+	-	Active set (1)
23	+	-	+	-	-	+	Active set (2)
24	+	-	+	-	-	-	Ruled out
25	+	-	-	+	+	+	Active set (1,2)
26	+	-	-	+	+	-	Ruled out
27	+	-	-	+	-	+	Active set (2)
28	+	-	-	+	-	-	Ruled out
29	+	-	-	-	+	+	Ruled out
30	+	-	-	-	+	-	Ruled out
31	+	-	-	-	-	+	Ruled out
32	+	-	-	-	-	-	Ruled out
33	-	+	+	+	+	+	Active set (1,2)
34	-	+	+	+	+	-	Ruled out
35	-	+	+	+	-	+	Ruled out
36	-	+	+	+	-	-	Ruled out
37	-	+	+	-	+	+	Active set (1,2)
38	-	+	+	-	+	-	Active set (1)
39	-	+	+	-	-	+	Ruled out
40	-	+	+	-	-	-	Ruled out
41	-	+	-	+	+	+	Ruled out
42	-	+	-	+	+	-	Ruled out
43	-	+	-	+	-	+	Ruled out
44	-	+	-	+	-	-	Ruled out
45	-	+	-	-	+	+	Ruled out
46	-	+	-	-	+	-	Ruled out
47	-	+	-	-	-	+	Ruled out
48	-	+	-	-	-	-	Ruled out
49	-	-	+	+	+	+	Ruled out
50	-	-	+	+	+	-	Ruled out
51	-	-	+	+	-	+	Ruled out
52	-	-	+	+	-	-	Ruled out
53	-	-	+	-	+	+	Ruled out
54	-	-	+	-	+	-	Ruled out
55	-	-	+	-	-	+	Ruled out
56	-	-	+	-	-	-	Ruled out
57	-	-	-	+	+	+	Ruled out

Case No.	$\lambda_{2,1}$	$\lambda_{3,1}$	$\lambda_{1,2}$	$\lambda_{3,2}$	$\lambda_{1,3}$	$\lambda_{2,3}$	Comments
58	-	-	-	+	+	-	Ruled out
59	-	-	-	+	-	+	Ruled out
60	-	-	-	+	-	-	Ruled out
61	-	-	-	-	+	+	Ruled out
62	-	-	-	-	+	-	Ruled out
63	-	-	-	-	-	+	Ruled out
64	-	-	-	-	-	-	Ruled out

Thus each case is either ruled out or corresponds to a desired active set.

II. $\lambda_1 > 0$, $\lambda_2 > 0$, $\lambda_3 < 0$, $\bar{\lambda}_1 > 0$, $\bar{\lambda}_2 < 0$ and $\bar{\lambda}_3 < 0$ (C2)

The possible active sets satisfying Lemma 2 are

(i) Constraints 1 and 2

(ii) Constraint 1

All possible cases are considered in the following table.

Case No.	$\lambda_{2,1}$	$\lambda_{3,1}$	$\lambda_{1,2}$	$\lambda_{3,2}$	$\lambda_{1,3}$	$\lambda_{2,3}$	Comments
1	+	+	+	+	+	+	Ruled out
2	+	+	+	+	+	-	Ruled out
3	+	+	+	+	-	+	Ruled out
4	+	+	+	+	-	-	Ruled out
5	+	+	+	-	+	+	Ruled out
6	+	+	+	-	+	-	Ruled out
7	+	+	+	-	-	+	Ruled out
8	+	+	+	-	-	-	Ruled out
9	+	+	-	+	+	+	Ruled out
10	+	+	-	+	+	-	Ruled out
11	+	+	-	+	-	+	Ruled out
12	+	+	-	+	-	-	Ruled out
13	+	+	-	-	+	+	Ruled out
14	+	+	-	-	+	-	Ruled out
15	+	+	-	-	-	+	Ruled out
16	+	+	-	-	-	-	Ruled out

Case No.	$\lambda_{2,1}$	$\lambda_{3,1}$	$\lambda_{1,2}$	$\lambda_{3,2}$	$\lambda_{1,3}$	$\lambda_{2,3}$	Comments
17	+	-	+	+	+	+	Active set (1,2)
18	+	-	+	+	+	-	Ruled out
19	+	-	+	+	-	+	Ruled out
20	+	-	+	+	-	-	Ruled out
21	+	-	+	-	+	+	Active set (1,2)
22	+	-	+	-	+	-	Active set (1)
23	+	-	+	-	-	+	Ruled out
24	+	-	+	-	-	-	Active set (1)
25	+	-	-	+	+	+	Ruled out
26	+	-	-	+	+	-	Ruled out
27	+	-	-	+	-	+	Ruled out
28	+	-	-	+	-	-	Ruled out
29	+	-	-	-	+	+	Active set (1,2)
30	+	-	-	-	+	-	Active set (1)
31	+	-	-	-	-	+	Ruled out
32	+	-	-	-	-	-	Active set (1)
33	-	+	+	+	+	+	Active set (1,2)
34	-	+	+	+	+	-	Ruled out
35	-	+	+	+	-	+	Ruled out
36	-	+	+	+	-	-	Ruled out
37	-	+	+	-	+	+	Active set (1,2)
38	-	+	+	-	+	-	Active set (1)
39	-	+	+	-	-	+	Ruled out
40	-	+	+	-	-	-	Ruled out
41	-	+	-	+	+	+	Ruled out
42	-	+	-	+	+	-	Ruled out
43	-	+	-	+	-	+	Ruled out
44	-	+	-	+	-	-	Ruled out
45	-	+	-	-	+	+	Ruled out
46	-	+	-	-	+	-	Ruled out
47	-	+	-	-	-	+	Ruled out
48	-	+	-	-	-	-	Ruled out
49	-	-	+	+	+	+	Active set (1,2)
50	-	-	+	+	+	-	Ruled out
51	-	-	+	+	-	+	Ruled out
52	-	-	+	-	-	-	Ruled out
53	-	-	+	-	+	+	Active set (1,2)
54	-	-	+	-	+	-	Active set (1)
55	-	-	+	-	-	+	Ruled out
56	-	-	+	-	-	-	Active set (1)
57	-	-	-	+	+	+	Ruled out
58	-	-	-	+	+	-	Ruled out
59	-	-	-	+	-	+	Ruled out
60	-	-	-	+	-	-	Ruled out
61	-	-	-	-	+	+	Ruled out
62	-	-	-	-	+	-	Ruled out

Case No.	$\lambda_{2,1}$	$\lambda_{3,1}$	$\lambda_{1,2}$	$\lambda_{3,2}$	$\lambda_{1,3}$	$\lambda_{2,3}$	Comments
63	-	-	-	-	-	+	Ruled out
64	-	-	-	-	-	-	Ruled out

Here, too, each case either leads to a desired active set or is ruled out.

APPENDIX D

```
C    THE PROGRAM MINIMIZES A CONVEX QUADRTIC FUNCTION SUBJECT TO LINEAR
C    INEQUALITY CONSTRAINTS.
C
      COMMON/STST/NUMBER
      COMMON/VAL/ AT(10,10),B(10),X(10),X0(10),A(10,10),SMQ(10)
      COMMON/VALUE/Q(10,10),QINV(10,10),ACTIV(10,10),GRAD(10),MM,ALAMBR(
     110),ALAM(10),STEP(10),BACTV(10)
      COMMON/SPACE/ TEMP1(10),TEMP2(10),TEMP3(10),DUM1(10,10),DUM2(10,10
     1),DUM3(10,10),WORK(150),IDGT
      COMMON/ACT/IACT(10)
      NUMBER=0
      IDGT=0
      N=3
      M=4
      M1=10
      WRITE(6,100)
  100 FORMAT(2X,'TEST PROBLEM NUMBER 3, CASE 8')
      CALL WRTMAT(A,N,M,M1,'A         ')
      CALL WRTVEC(B,M,'B         ')
      CALL WRTVEC(SMQ,N,'SMQ       ')
      CALL WRTVEC(X0,N,'X0        ')
      CALL WRTMAT(Q,N,N,M1,'Q         ')
      DO 11 I=1,N
      DO 11 J=1,N
   11 DUM1(I,J)=Q(I,J)
      CALL LINV2F(DUM1,N,M1,QINV,IDGT,WORK,IER)
      CALL WRTMAT(QINV,N,N,M1,'QINV      ')
      CALL TRANSP(A,AT,N,M,M1,M1)
      DO 4 III=1,4
      X0(1)=3.
      X0(2)=0.
      X0(3)=0.
      CALL MATVEC(AT,X0,TEMP1,M,N,M1)
      MM=0
      CALL WRTVEC(TEMP1,M,'TEMP1     ')
      DO 1 I=1,M
      DUM=ABS(TEMP1(I)-B(I))
      IF(DUM.GT..001) GO TO 1
      MM=MM+1
      IACT(MM)=I
      BACTV(MM)=B(I)
      DO 2 J=1,N
    2 ACTIV(J,MM)=A(J,I)
    1 CONTINUE
      DO 5 I=1,N
  5 X(I)=XO(I)
    CALL WRTMAT(ACTIV,N,MM,M1,'ACTIV   ')
    CALL WRTVEC(BACTV,MM,'BACTV   ')
    CALL WRTVEI(IACT,MM,'IACT    ')
    IF(III.EQ.4) GO TO 7


    IF(III.EQ.3) GO TO 6
    IF(III.EQ.2) GO TO 3
    WRITE(6,110)
110 FORMAT(//,2X,'USING FLETCHER S METHOD')
    CALL FLETER(N,M,M1)
    GO TO 4
  3 NUMBER=0
    WRITE(6,111)
111 FORMAT(//,2X,'USING NEW METHOD')
    CALL NEWLGC(N,M,M1)
    GO TO 4
  6 NUMBER=0
    WRITE(6,112)
112 FORMAT(//,2X,'USING SRAGENT S METHOD')
    CALL SARGAN(N,M,M1)
    GO TO 4
  7 NUMBER=0
    WRITE(6,113)
113 FORMAT(//,2X,'USING ROSEN S METHOD')
    CALL ROSEN(N,M,M1)
  4 WRITE(6,10) NUMBER
 10 FORMAT(2X,'NUMBER OF GRADIENT EVALUATIONS=',I3)
    STOP
    END
```

```
      BLOCK DATA
      COMMON/VAL/AT(10,10),B(10),X(10),XO(10),A(10,10),SMQ(10)
      COMMON/VALUE/Q(10,10),QINV(10,10),ACTIV(10),GRAD(10),MM,ALAMBR(10)
     1,ALAM(10),STEP(10),BACTV(10)
      DATA SMQ/-8.,-6.,-4.,7*0./
      DATA Q/4.,2.,2.,7*0.,2.,4.,8*0.,2.,0.,2.,7*0.,70*0./
      DATA A/-1.,-1.,-2.,7*0.,1.,9*0.,0.,1.,8*0.,0.,0.,1.,7*0.,60*0./
      DATA B/-3.,9*0./
      DATA XO/3.,0.,0.,0.,6*0./
      END
```

```
C     SUBROUTINE FLETCH DETERMINES THE ACTIVE SET USING THE ACTIVE CONS-
C     TRAINT LOGIC PROPOSED BY FLETCHER.
C
      SUBROUTINE FLETER(N,M,M1)
      COMMON/VALUE/Q(10,10),QINV(10,10),ACTIV(10,10),GRAD(10),MM,ALAMBR(
     110),ALAM(10),STEP(10),BACTV(10)
      COMMON/SPACE/TEMP1(10),TEMP2(10),TEMP3(10),DUM1(10,10),DUM2(10,10)
     1,DUM3(10,10),WORK(150),IDGT
      COMMON/VAL/AT(10,10),B(10),X(10),X0(10),A(10,10),SMQ(10)
      COMMON/ACT/IACT(10)
      DIMENSION IORD(10),IORR(10)
      WRITE(6,200)
  200 FORMAT(2X,'FLETCHER     ')
    5 CALL GRADNT(N,M,M1)
    8 CALL DIRECT(N,M,M1)
      PNORM=0.
      DO 1 I=1,N
    1 PNORM=PNORM+STEP(I)**2
      IF(PNORM.GT.1.0E-4) GO TO 2
      IF(MM.EQ.0) GO TO 7
      CALL MINMAX(ALAM,MM,XMIN,IDX1,XMAX,IDX2)
      IF(XMIN.LT.0.) GO TO 3
      GO TO 7
    3 MM=MM-1
      DO 4 J=IDX1,MM
      J1=J+1
      IACT(J)=IACT(J1)
      BACTV(J)=BACTV(J1)
      DO 4 I=1,N
    4 ACTIV(I,J)=ACTIV(I,J1)
      GO TO 8
    2 DO 6 I=1,N
      ADUM=X(I)
      X(I)=X(I)+STEP(I)
    6 X0(I)=ADUM
      CALL STSFY(N,M,M1)
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      GO TO 5
    7 RETURN
      END
```

```
C     SUBROUTINE SARGAN DETERMINES THE ACTIVE SET USING SARGENT'S METHOD.
C
      SUBROUTINE SARGAN(N,M,M1)
      COMMON/VALUE/Q(10,10),QINV(10,10),ACTIV(10,10),GRAD(10),MM,ALAMBR(
     110),ALAM(10),STEP(10),BACTV(10)
      COMMON/VAL/AT(10,10),B(10),X(10),X0(10),A(10,10),SMQ(10)
      COMMON/ACT/IACT(10)
      COMMON/SPACE/TEMP1(10),TEMP2(10),TEMP3(10),DUM1(10,10),DUM2(10,10)
     1,DUM3(10,10),WORK(150),IDGT
      DIMENSION IORD(10),IKP(10),IOLD(10)
      MK=0
    3 CALL GRADNT(N,M,M1)
      CALL DIRECT(N,M,M1)
      PNORM=0.
      DO 1 I=1,N
    1 PNORM=PNORM+STEP(I)**2
      IF(PNORM.GT.1.0E-4) GO TO 2
      CALL MINMAX(ALAM,MM,XMIN,IDX1,XMAX,IDX2)
      IF(XMIN.GT.0.) GO TO 12
      MK=0
    6 CALL MINMAX(ALAM,MM,XMIN,IDX1,XMAX,IDX2)
      IF(XMIN.GT.0.) GO TO 4
      MK=MK+1
      IKP(MK)=IACT(IDX1)
      MM=MM-1
      DO 5 I=IDX1,MM
      I1=I+1
    5 IACT(I)=IACT(I1)
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      CALL DIRECT(N,M,M1)
      IF(MM.EQ.0) GO TO 4
      GO TO 6
    2 CALL ORDER(ALAM,IACT,IORD,MM)
      IF(MK.GT.0) GO TO 14
      J=1
      GO TO 9
   14 DO 7 J=1,MM
      DO 8 I=1,MK
      IF(IORD(J).NE.IKP(I)) GO TO 8
      GO TO 7
    8 CONTINUE
      GO TO 9
    7 CONTINUE
      GO TO 4
    9 IT=IORD(J)
      DO 15 K=1,MM
   15 IF(IACT(K).EQ.IORD(J)) GO TO 16
```

```
16 MM=MM-1
   IF(ALAM(K).GT.0.) GO TO 4
   MK=MK+1



   IKP(MK)=IT
   IF(MM.EQ.0) GO TO 4
   DO 10 I=K,MM
   I1=I+1
10 IACT(I)=IACT(I1)
   CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
   CALL DIRECT(N,M,M1)
   GO TO 2
 4 DO 13 I=1,N
   ADUM=X(I)
   X(I)=X(I)+STEP(I)
13 X0(I)=ADUM
   CALL STSFY(N,M,M1)
   PNORM=0.
   MOLD=MM
   DO 17 I=1,N
   IOLD(I)=IACT(I)
17 PNORM=PNORM+(X0(I)-X(I))**2
   IF(PNORM.GT..001) GO TO 20
   CALL MATVEC(AT,STEP,TEMP1,M,N,M1,M1)
   IDUM=1
   II=1
   MK=0
   I1=IOLD(MOLD)
   DO 19 I=1,M
   IF(IDUM.GT.I1) GO TO 18
   IF(IOLD(IDUM).NE.I) GO TO 19
   IDUM=IDUM+1
   IF(TEMP1(I).LT..001) GO TO 22
   MK=MK+1
   IKP(MK)=I
   GO TO 19
22 IACT(II)=I
   II=II+1
19 CONTINUE
18 MM=II-1
20 CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
   GO TO 3
12 RETURN
   END
```

```
      SUBROUTINE ROSEN(N,M,M1)
      COMMON/VALUE/Q(10,10),QINV(10,10),ACTIV(10,10),GRAD(10),MM,ALAMBR(
     110),ALAM(10),STEP(10),BACTV(10)
      COMMON/VAL/AT(10,10),B(10),X(10),XO(10),A(10,10),SMQ(10)
      COMMON/ACT/IACT(10)
      COMMON/SPACE/TEMP1(10),TEMP2(10),TEMP3(10),DUM1(10,10),DUM2(10,10)
     1,DUM3(10,10),WORK(150),IDGT
      COMMON/OK/OK(10)
      IA=0
   13 CALL GRADNT(N,M,M1)
    4 CALL DIRECT(N,M,M1)
      PNORM=0.
      DO 1 I=1,N
    1 PNORM=PNORM+STEP(I)**2
      IF(PNORM.GT.1.0E-4) GO TO 5
   10 CALL MINMAX(ALAM,MM,XMIN,IDX1,XMAX,IDX2)
      IF(XMIN.GT.0.) GO TO 14
      DO 2 I=1,MM
      DUM=ABS(ALAM(I))
    2 TEMP1(I)=(ALAM(I)**2)*ALAM(I)/(OK(I)*DUM)
      CALL MINMAX(TEMP1,MM,XMIN,IDX1,XMAX,IDX2)
      MM=MM-1
      DO 3 I=IDX1,MM
      I1=I+1
    3 IACT(I)=IACT(I1)
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      IA=1
      GO TO 4
    5 IF(IA.GT.0) GO TO 11
      DO 6 I=1,MM
      DUM=ABS(ALAM(I))
    6 TEMP1(I)=(ALAM(I)**2)*ALAM(I)/(OK(I)*DUM)
      CALL MINMAX(TEMP1,MM,XMIN,IDX1,XMAX,IDX2)
      DUM=0.
      DO 7 I=1,N
    7 DUM=DUM+STEP(I)*GRAD(I)
      DUM=-.5*DUM
      XMAX=-XMIN
      IF(XMAX.LE.DUM) GO TO 11
      MM=MM-1
      DO 8 I=IDX1,MM
      I1=I+1
    8 IACT(I)=IACT(I1)
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      CALL DIRECT(N,M,M1)
      PNORM=0.
      DO 9 I=1,N
    9 PNORM=PNORM+STEP(I)**2
      IF(PNORM.GT.1.0E-4) GO TO 11
      GO TO 10



   11 DO 12 I=1,N
      ADUM=X(I)
      X(I)=X(I)+STEP(I)
   12 XO(I)=ADUM
      CALL STSFY(N,M,M1)
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      IA=0
      GO TO 13
   14 RETURN
      END
```

```
C    SUBROUTINE NEWLGC DETERMINES THE ACTIVE SET USING THE NEW ACTIVE
C    CONSTRAINT LOGIC.
C
      SUBROUTINE NEWLGC(N,M,M1)
      COMMON/VALUE/Q(10,10),QINV(10,10),ACTIV(10,10),GRAD(10),MM,ALAMBR(
     110),ALAM(10),STEP(10),BACTV(10)
      COMMON/VAL/AT(10,10),B(10),X(10),X0(10),A(10,10),SMQ(10)
      COMMON/ACT/IACT(10)
      COMMON/SPACE/TEMP1(10),TEMP2(10),TEMP3(10),DUM1(10,10),DUM2(10,10)
     1,DUM3(10,10),WORK(150),IDGT
      DIMENSION IMAY(10)
    5 CALL GRADNT(N,M,M1)
      CALL DIRECT(N,M,M1)
      IF(MM.LE.2) GO TO 8
      CALL MINMAX(ALAM,MM,XMIN,IDX1,XMAX,IDX2)
      IF(XMIN.GT.0.) GO TO 3
      CALL MINMAX(ALAMBR,MM,XMIN,IDX1,XMAX,IDX2)
      IF(XMAX.GT.0.) GO TO 31
      MM=0
      GO TO 3
   31 IF(MM.EQ.3) GO TO 9
      MC=0
      DUMMY=0.
      DO 32 I=1,MM
      IF(ALAM(I).LT.0.) GO TO 40
      MC=MC+1
      IMAY(MC)=IACT(I)
      GO TO 32
   40 IF(ALAM(I).GT.DUMMY) GO TO 32
      DUMMY=ALAM(I)
      ID=IACT(I)
   32 CONTINUE
      CALL WRTVEI(IMAY,MC,'IMAY    ')
   39 DO 33 I=1,MM
      IF(IACT(I).LE.ID) GO TO 33
      I1=I-1
      IACT(I1)=IACT(I)
   33 CONTINUE
      MM=MM-1
      ID=0
      IF(MM.EQ.0) GO TO 3
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      IF(MC.EQ.0) GO TO 3
      CALL DIRECT(N,M,M1)
      CALL MINMAX(ALAM,MM,XMIN,IDX1,XMAX,IDX2)
      IF(ALAM(IDX1).GT.0.) GO TO 3
      DUMMY=0.
      MP=0
      DO 35 I=1,MC
      DO 36 J=1,MM



      IF(IACT(J).LT.IMAY(I)) GO TO 36
      IC=J
      GO TO 37
   36 CONTINUE
   37 IF(ALAM(IC).LT.0.) GO TO 38
      MP=MP+1
      IMAY(MP)=IMAY(I)
      GO TO 35
   38 IF(ALAM(IC).GT.DUMMY) GO TO 35
      DUMMY=ALAM(IC)
      ID=IC
   35 CONTINUE
      MC=MP
      CALL WRTVEI(IMAY,MC,'IMAY    ')
      IF(ID.EQ.0) GO TO 3
      GO TO 39
    9 ABSOL=ABS(ALAMBR(IDX1))
      DO 11 I=1,MM
      IF(I.EQ.IDX1) GO TO 11
      IF(I.EQ.IDX2) GO TO 11
      I1=I
   11 CONTINUE
      IF(ABSOL.LT.ALAMBR(IDX2)) GO TO 10
      CALL MULTPR(IDX2,I1,N,M,M1)
      IF((TEMP1(1).GT.0.).AND.(TEMP1(2).GT.0.)) GO TO 12
      MM=1
      IACT(1)=IACT(IDX2)
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      GO TO 3
   12 IA=IACT(IDX2)
      IB=IACT(I1)
   15 MM=2
      IF(IA.LT.IB) GO TO 13
      IACT(1)=IB
      IACT(2)=IA
      GO TO 14
   13 IACT(1)=IA
```

```
      IACT(2)=IB
   14 CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      GO TO 3
   10 IF((ALAM(IDX1).GT.0.).AND.(ALAM(I1).GT.0.)) GO TO 16
      GO TO 17
   16 IA=IACT(IDX1)
      IB=IACT(I1)
      GO TO 15
   17 IF((ALAM(IDX1).GT.0.).AND.(ALAM(I1).LT.0.)) GO TO 18
      GO TO 19
   18 CALL MULTPR(IDX1,IDX2,N,M,M1)
      IF(TEMP1(1).LT.0.) GO TO 20
      IA=IACT(IDX1)
      IB=IACT(IDX2)
      GO TO 15
   20 MM=1
      IACT(1)=IACT(IDX2)
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)


      GO TO 3
   19 IF((ALAM(IDX1).LT.0.).AND.(ALAM(I1).GT.0.)) GO TO 26
      GO TO 28
   26 CALL MULTPR(I1,IDX2,N,M,M1)
      IF(TEMP1(1).LT.0.) GO TO 27
      IA=IACT(I1)
      IB=IACT(IDX2)
      GO TO 15
   27 MM=1
      IACT(1)=IACT(IDX2)
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      GO TO 3
   28 CALL MULTPR(IDX1,IDX2,N,M,M1)
      IF(TEMP1(1).LT.0.) GO TO 21
      IA=IACT(IDX1)
      IB=IACT(IDX2)
      GO TO 15
   21 CALL MULTPR(I1,IDX2,N,M,M1)
      IF(TEMP1(1).LT.0.) GO TO 22
      IA=IACT(I1)
      IB=IACT(IDX2)
      GO TO 15
   22 MM=1
      IACT(1)=IACT(IDX2)
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      GO TO 3
    8 IF(MM.LE.1) GO TO 4
      CALL MINMAX(ALAM,MM,XMIN,IDX1,XMAX,IDX2)
      IF(XMIN.LT.0.) GO TO 1
      GO TO 3
    1 IF(ALAMBR(IDX2).LT.0.) GO TO 2
      MM=1
      IACT(1)=IACT(IDX2)
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      GO TO 3
    2 MM=0
      GO TO 3
    4 IF(MM.EQ.0) GO TO 3
      IF(ALAM(1).GT.0.) GO TO 3
      MM=0
    3 CALL DIRECT(N,M,M1)
      PNORM=0.
      DO 7 I=1,N
    7 PNORM=PNORM+STEP(I)**2
      IF(PNORM.LT.1.0E-4) GO TO 6
      DO 25 I=1,N
      ADUM=X(I)
      X(I)=X(I)+STEP(I)
   25 XO(I)=ADUM
      CALL STSFY(N,M,M1)
      CALL ACTSET(A,B,ACTIV,BACTV,MM,N)
      GO TO 5
    6 RETURN
      END
```

```
C     SUBROUTINE STSFY CALCULATES THE REQUIRED STEP SIZE IN THE GIVEN
C     STEP DIRECTION. IT GIVES THE NEW VALUE OF X.
C
      SUBROUTINE STSFY(N,M,M1)
      COMMON/VAL/ AT(10,10),B(10),X(10),X0(10),A(10,10),SMQ(10)
      COMMON/VALUE/ Q(10,10),QINV(10,10),ACTIV(10,10),GRAD(10),MM,ALAMBR
     1(10),ALAM(10),STEP(10),BACTV(10)
      COMMON/SPACE/TEMP1(10),TEMP2(10),TEMP3(10),DUM1(10,10),DUM2(10,10)
     1,DUM3(10,10),WORK(150),IDGT
      COMMON/ACT/IACT(10)
      WRITE(6,200)
  200 FORMAT(2X,'STSFY   ')
      IF(MM.EQ.M) RETURN
    3 MTRBLE=0
      CALL MATVEC(AT,X,TEMP1,M,N,M1,M1)
      CALL WRTVEC(X,N,'X         ')
      CALL WRTVEC(TEMP1,M,'TEMP1    ')
      DO 1 I=1,M
      TEMP2(I)=TEMP1(I)-B(I)
      WRITE(6,222)TEMP2(I)
  222 FORMAT(2X,F12.4)
      IF(TEMP2(I).GT.-.01) GO TO 1
      MTRBLE=I
      GO TO 2
    1 CONTINUE
      IF(MTRBLE.EQ.0) GO TO 6
    2 AP=0.
      AX=0.
      DO 4 J=1,N
      AX=AX+X(J)*A(J,MTRBLE)
    4 AP=AP+STEP(J)*A(J,MTRBLE)
      WRITE(6,333)MTRBLE,MO
  333 FORMAT(2X,'MTRBLE=',I5,5X,'MO=',I5)
      WRITE(6,111) AP,AX
  111 FORMAT(2X,'AP= ',F12.4,10X,'AX= ',F12.4)
      THTA=(B(MTRBLE)-AX)/AP
      DO 5 J=1,N
    5 X(J)=X(J)+THTA*STEP(J)
      CALL WRTVEC(X,N,'X        ')
      GO TO 3
    6 CALL MATVEC(AT,X,TEMP1,M,N,M1,M1)
      CALL VECSUB(TEMP1,B,TEMP2,M,M1)
      II=0
      DO 7 I=1,M
      IF(TEMP2(I).GT..001) GO TO 7
      II=II+1
      IACT(II)=I
    7 CONTINUE
      MM=II
      CALL WRTVEI(IACT,MM,'IACT     ')


      RETURN
      END
```

```
C    SUBROUTINE DIRECT CALCULATES THE MULTIPLIERS(LAMBDA AND LAMBDABAR)
C    AND THE REQUIRED STEP SIZE TO REACH THE MINIMUN WITH THE CONSTRAIN-
C    TS IN THE ACTIVE SET TAKEN AS EQUALITIES.
C
      SUBROUTINE DIRECT(N,M,M1)
      COMMON/VALUE/ Q(10,10),QINV(10,10),ACTIV(10,10),GRAD(10),MM,ALAMBR
     1(10),ALAM(10),STEP(10),BACTV(10)
      COMMON/ACT/IACT(10)
      COMMON/VAL/AT(10,10),B(10),X(10),XO(10),A(10,10),SMQ(10)
      COMMON/SPACE/ TEMP1(10),TEMP2(10),TEMP3(10),DUM1(10,10),DUM2(10,10
     1),DUM3(10,10),WORK(150),IDGT
      COMMON/OK/OK(10)
      DIMENSION DUMMY1(10,10),DUMMY2(10,10),TEMP(10)
      IF(MM.GT.0) GO TO 9
      CALL MATVEC(QINV,SMQ,TEMP1,N,N,M1,M1)
      DO 10 I=1,N
   10 STEP(I)=-TEMP1(I)-X(I)
      GO TO 20
    9 IDGT=0
      DO 1 I=1,MM
      CAS=0.
      DO 2 J=1,N
    2 CAS=ACTIV(J,I)**2+CAS
      DO 3 K=1,N
    3 DUMMY1(K,I)=ACTIV(K,I)/CAS
    1 CONTINUE
      CALL TRANSP(DUMMY1,DUMMY2,N,MM,M1,M1)
      CALL MULT(DUMMY2,QINV,DUM2,MM,N,N,M1,M1,M1)
      CALL MULT(DUM2,DUMMY1,DUM3,MM,N,MM,M1,M1,M1)
      CALL LINV2F(DUM3,MM,M1,DUM1,IDGT,WORK,IER)
      DO 4 I=1,MM
    4 OK(I)=DUM1(I,I)
      CALL MATVEC(DUM2,GRAD,ALAMBR,MM,N,M1,M1)
      CALL MATVEC(DUM1,ALAMBR,ALAM,MM,MM,M1,M1)
      CALL TRANSP(ACTIV,DUM1,N,MM,M1,M1)
      CALL MULT(DUM1,QINV,DUM2,MM,N,N,M1,M1,M1)
      CALL MULT(DUM2,ACTIV,DUM3,MM,N,MM,M1,M1,M1)
      IDGT=0
      CALL LINV2F(DUM3,MM,M1,DUM1,IDGT,WORK,IER)
      CALL MATVEC(DUM2,GRAD,TEMP3,MM,N,M1,M1)
      CALL WRTVEC(ALAMBR,MM,'ALAMBR  ')
      CALL MATVEC(DUM1,TEMP3,TEMP,MM,MM,M1,M1)
      CALL WRTVEC(ALAM,MM,'ALAM    ')
      CALL MATVEC(ACTIV,TEMP,TEMP1,N,MM,M1,M1)
      CALL VECSUB(TEMP1,GRAD,TEMP2,N,M1)
      CALL MATVEC(QINV,TEMP2,STEP,N,N,M1,M1)
      CALL WRTVEC(STEP,N,'STEP     ')
   20 CALL WRTVEI(IACT,MM,'IACT     ')
      RETURN
      END
```

```
C    GIVEN THE VECTORS IN THE ACTIVE SET, SUBROUTINE ACTSET DETERMINES
C    THE CORRESPONDING ACTIV AND BACTV.
C
      SUBROUTINE ACTSET(A,B,ACTIV,BACTV,MM,N)
      COMMON/ACT/IACT(10)
      DIMENSION A(10,10),BACTV(10),ACTIV(10,10),B(10)
      DO 1 I=1,MM
      K=IACT(I)
      BACTV(I)=B(K)
      DO 1 J=1,N
    1 ACTIV(J,I)=A(J,K)
      RETURN
      END
```

```
C     GIVEN VECTORS X AND IIN OF ORDER N, SUBROUTINE ORDER REARRANGES
C     THE ELEMENTS OF IIN SUCH THAT IN THE NEW ARRANGEMENT THE CORRESOPN-
C     DING ELEMENTS OF X ARE IN ASCENDING ORDER. THIS IS RETURNED AS VEC-
C     TOR IORD.
C
      SUBROUTINE ORDER(X,IIN,IORD,N)
      DIMENSION IIN(1),IORD(1),DUM(7),X(1),IDUM(7)
      DO 1 I=1,N
      DUM(I)=X(I)
    1 IDUM(I)=IIN(I)
      NN=N
      DO 2 I=1,N
      CALL MINMAX(DUM,NN,XMIN,IDX1,XMAX,IDX2)
      IORD(I)=IDUM(IDX1)
      NN=NN-1
      IF(NN.EQ.0) GO TO 4
      DO 3 J=IDX1,NN
      J1=J+1
      DUM(J)=DUM(J1)
    3 IDUM(J)=IDUM(J1)
      GO TO 2
    4 J1=IDX1+1
      IDUM(IDX1)=IDUM(J1)
      DUM(IDX1)=DUM(J1)
    2 CONTINUE
      RETURN
      END
```

```
C    SUBROUTINE MINMAX LOCATES THE MINIMUM AND MAXIMUM COMPONENTS OF
C    THE VECTOR X.
C
      SUBROUTINE MINMAX(X,N,XMIN,IDX1,XMAX,IDX2)
      COMMON/ACT/IACT(10)
      DIMENSION X(1)
      XMIN=X(1)
      XMAX=X(1)
      IDX1=1
      IDX2=1
      IF(N.LE.1) RETURN
      DO 10 I=2,N
      IF(XMAX.GE.X(I)) GO TO 1
      XMAX=X(I)
      IDX2=I
    1 IF(XMIN.LE.X(I)) GO TO 10
      XMIN=X(I)
      IDX1=I
   10 CONTINUE
      RETURN
      END
```

```
C     SUBROUTINE MULTPR CALCULATES THE MULTIPLIERS FOR THE CASE M=2.
C
      SUBROUTINE MULTPR(I1,I2,N,M,M1)
      COMMON/VAL/AT(10,10),B(10),X(10),X0(10),A(10,10),SMQ(10)
      COMMON/VALUE/Q(10,10),QINV(10,10),ACTIV(10,10),GRAD(10),MM,ALAMBR(
     110),ALAM(10),STEP(10),BACTV(10)
      COMMON/SPACE/TEMP1(10),TEMP2(10),TEMP3(10),DUM1(10,10),DUM2(10,10)
     1,DUM3(10,10),WORK(150),IDGT
      COMMON/ACT/IACT(10)
      IDGT=0
      WRITE(6,10) IACT(I1),IACT(I2)
   10 FORMAT(5X,'I1=',I2,10X,'I2=',I2)
      I11=IACT(I1)
      I12=IACT(I2)
      DO 1 J=1,N
      DUM1(J,1)=A(J,I11)
    1 DUM1(J,2)=A(J,I12)
      CALL TRANSP(DUM1,DUM2,N,2,M1,M1)
      CALL MULT(DUM2,QINV,DUM3,2,N,N,M1,M1,M1)
      CALL MULT(DUM3,DUM1,DUM2,2,N,2,M1,M1,M1)
      CALL LINV2F(DUM2,2,M1,DUM1,IDGT,WORK,IER)
      CALL MULT(DUM1,DUM3,DUM2,2,2,N,M1,M1,M1)
      CALL MATVEC(DUM2,GRAD,TEMP1,2,N,M1,M1)
      CALL WRTVEC(TEMP1,2,'MULTPR  ')
      RETURN
      END
```

```
C  SUPPORT SUBROUTINES
C
C
      SUBROUTINE WRTMAT(A,N,M,IA,ANAME)
      DIMENSION A(IA,1)
      REAL*8 ANAME
      WRITE(6,100)ANAME,N,M
  100 FORMAT(/,' MATRIX ',A8,3X,'(',I3,' ROWS X',I3,' COLS)')
      IF(M.LE.10) GO TO 15
      DO 10 I=1,N
   10 WRITE(6,101)(A(I,J),J=1,M)
  101 FORMAT(/,(1P10E13.5))
      RETURN
   15 DO 20 I=1,N
   20 WRITE(6,102)(A(I,J),J=1,M)
  102 FORMAT(1P10E13.5)
      RETURN
      END
C
C
C
      SUBROUTINE MULT(A,B,C,N,L,M,NA,NB,NC)
      DIMENSION A(NA,1),B(NB,1),C(NC,1)
      DO 20 I=1,N
      DO 20 J=1,M
      TEMP=0.
      DO 10 K=1,L
   10 TEMP=TEMP+A(I,K)*B(K,J)
   20 C(I,J)=TEMP
      RETURN
      END
C
C
C
      SUBROUTINE MATSUB(A,B,C,N,M,NA,NB,NC)
      DIMENSION A(NA,1),B(NB,1),C(NC,1)
      DO 10 I=1,N
      DO 10 J=1,M
   10 C(I,J)=A(I,J)-B(I,J)
      RETURN
      END
C
C
C
      SUBROUTINE VECSUB(A,B,C,N,NA)
      DIMENSION A(NA),B(NA),C(NA)
      DO 10 I=1,N
   10 C(I)=A(I)-B(I)
      RETURN
      END



C
C
C
      SUBROUTINE MATVEC(A,B,C,N,M,NA,MA)
      DIMENSION A(NA,1),B(MA),C(NA)
      DO 10 I=1,N
      TEMP=0.
      DO 20 J=1,M
   20 TEMP=TEMP+A(I,J)*B(J)
   10 C(I)=TEMP
      RETURN
      END
C
C
C
      SUBROUTINE TRANSP(A,AT,N,M,NA,MA)
      DIMENSION A(NA,1),AT(MA,1)
      DO 10 I=1,N
      DO 10 J=1,M
   10 AT(J,I)=A(I,J)
      RETURN
      END
C
C
C
      SUBROUTINE WRTVEC(A,N,ANAME)
      DIMENSION A(1)
      REAL*8 ANAME
      WRITE(6,100) ANAME,N
  100 FORMAT(/,' VECTOR ',A8,' OF DIMENSION ',I3)
      WRITE(6,101)(A(I),I=1,N)
  101 FORMAT(1P10E13.5)
      RETURN
      END
C
C
C
```

```
      SUBROUTINE WRTVEI(A,N,ANAME)
      DIMENSION A(1)
      REAL*8 ANAME
      WRITE(6,100) ANAME,N
  100 FORMAT(/,' VECTOR ',A8,' OF DIMENSION ',I3)
      WRITE(6,101)(A(I),I=1,N)
  101 FORMAT(10(3X,I7))
      RETURN
      END
```

TABLE 1. TEST EXAMPLE #1 (CASES 1, 2 and 3)

Objective function

$$f(x) = .01x_1^2 + x_2^2 - 100.$$

Constraints

$$10x_1 - x_2 \geq 10.$$

$$x_1 \geq 2.$$

$$-x_1 \geq -50.$$

$$x_2 \geq -50.$$

$$-x_2 \geq -50.$$

Case 1. Starting point (Constraints 1 and 2 satisfied as equalities)

$x0 = (2.0 , 10.0)$

	Fletcher	Sargent	Rosen	New Logic
No. of gradient evaluations	2	2	2	2

TABLE 1. (CONTINUED)

Case 2. Starting point (Constraints 1 and 5 satisfied as equalities)

xo = (6.0 , 50.0)

	Fletcher	Sargent	Rosen	New Logic
No. of gradient evaluations	5	3	3	3

Case 3. Starting point (Constraints 3 and 5 satisfied as equalities)

xo = (50.0 , 50.0)

	Fletcher	Sargent	Rosen	New Logic
No. of gradient evaluations	3	3	3	3

TABLE 2. TEST EXAMPLE #2 (CASES 4, 5 AND 6)

Objective function

$$f(x) = x_1^2 + .5x_2^2 + x_3^2 - x_1x_3 + x_3x_4 - x_1 - 3x_2 + x_3 - x_4 + .5x_4^2$$

Constraints

$$-x_1 - 2x_2 - x_3 - x_4 \geq -5.$$
$$-3x_1 - x_2 - 2x_3 + x_4 \geq -4.$$
$$x_2 + 4x_3 \geq 1.5$$
$$x_1 \geq 0.$$
$$x_2 \geq 0.$$
$$x_3 \geq 0.$$
$$x_4 \geq 0.$$

TABLE 2. (CONTINUED)

Case 4. Starting point (Interior point)

xo = (.5 , .5 , .5 , .5)

	Fletcher	Sargent	Rosen	New Logic
No. of gradient evaluations	6	5	5	5

Case 5. Starting point (Constraints 1, 2 and 3 satisfied as equalities)

xo = (1.4210526 , 0.9736842 , 0.1315789 , 1.5)

	Fletcher	Sargent	Rosen	New Logic
No. of gradient evaluations	4	3	4	3

TABLE 2. (CONTINUED)

Case 6. Starting point (Constraints 3, 4, 6 and 7 satisfied as equalities)

xo = (0.0 , 1.5 , 0.0 , 0.0)

	Fletcher	Sargent	Rosen	New Logic
No. of gradient evaluations	4	4	4	4

TABLE 3. TEST EXAMPLE #3 (CASES 7, 8 AND 9)

Objective function

$$f(x) = 2x_1^2 + 2x_2^2 + x_3^2 + 2x_1x_2 + 2x_1x_3 - 8x_1 - 6x_2 - 4x_3 + 9$$

Constraints

$$-x_1 - x_2 - 2x_3 \geq -3.$$
$$x_1 \geq 0.$$
$$x_2 \geq 0.$$
$$x_3 \geq 0.$$

Case 7. Starting point (Interior point)

xo = (0.5 , 0.5 , 0.5)

	Fletcher	Sargent	Rosen	New Logic
No. of gradient evaluations	3	3	3	3

TABLE 3. (CONTINUED)

Case 8. Starting point (Constraints 1, 3 and 4 satisfied as equalities)

xo = (3.0 , 0.0 , 0.0)

	Fletcher	Sargent	Rosen	New Logic
No. of gradient evaluations	5	3	4	2

Case 9. Starting point (Constraints 2, 3 and 4 satisfied as equalities)

xo = (0.0 , 0.0 , 0.0)

	Fletcher	Sragent	Rosen	New Logic
No. of gradient evaluations	5	3	5	3

TABLE 4. TEST EXAMPLE #4 (CASES 10, 11 AND 12)

Objective function

$$f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

Constraints

$$- 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq -105.$$

$$- 10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0.$$

$$8x_1 - 2x_2 - 5x_9 + 2x_{10} \geq -12.$$

$$- 3x_1 - 4x_2 - 2x_3 + 7x_4 \geq -138.$$

$$- 5x_1 - 8x_2 - x_3 + 2x_4 \geq -46.$$

$$- .5x_1 - 2x_2 - 3x_5 + x_6 \geq -42.$$

$$- x_1 - 2x_2 - 14x_5 + 6x_6 \geq -20.$$

$$3x_1 - 6x_2 - 12x_9 + 7x_{10} \geq -96.$$

TABLE 4 (CONTINUED)

Case 10. Starting point (Interior point)

$xo = (2, 3, 5, 5, 1, 2, 7, 3, 6, 10)$

	Fletcher	Sargent	Rosen	New Logic
No. of gradient evaluations	8	7	7	7

Case 11. Starting point (Constraints 2, 6 and 7 satisfied as equalities)

$xo = (0, 0, 0, 0, 58, 132, 0, 0, 0, 0)$

	Fletcher	Sargent	Rosen	New Logic
No. of gradient evaluations	6	6	6	5

TABLE 4 (CONTINUED)

Case 12. Starting point (All constraints satisfied as equalities)

xo = (0. , 0. , 15.333333 , -15.333333 , 58. , 132. , 1.4285714 , 12.142856 , -9.8181818 , -30.545454)

	Fletcher	Sargent	Rosen	New Logic
No. of gradient evaluations	6	2	5	5

AN ACTIVE-CONSTRAINT LOGIC FOR NONLINEAR PROGRAMMING

by

Alok Das

(ABSTRACT)

The choice of active-constraint logic for screening inequalities has a major impact on the performance of gradient-projection method. It has been found that least-constrained strategies, which keep the number of constraints in the active set as small as possible, are computationally most efficient. However, these strategies are often prone to cycling of constraints between active and inactive status. This occurs mainly due to the violation of some of the constraints, taken as inactive, by the resulting step. This research develops methods for choosing an active set such that constraints in the active set satisfy the Kuhn-Tucker conditions and the resulting step does not violate the linear approximations to any of the constraints satisfied as equalities but considered inactive.

Some of the existing active-constraint logics, specifically the dual-violator rule, yield the desired active set when two constraints are satisfied as equalities. However, when three or more constraints are satisfied as equalities, none of the existing logics give the desired active set.

A number of general results, which help in the selection of the active set, have been developed in this research. An active-constraint logic has been developed for the case of three constraints. This logic gives the desired active-set. For the general case, when more than three constraints are satisfied as equalities, a separate active-set logic is suggested. This guarantees the nonviolation of the linear approximations to any of the constraints, taken as inactive, by the resulting step. The resulting active-set may not, however, satisfy the Kuhn-Tucker conditions.

The efficiency of the proposed logic was tested computationally using quadratic programming problems. Three existing active-set strategies were used for comparision. The proposed logic almost always performed as well or better than the best among the three existing active-set strategies.